MASTER EN SCIENCES INFORMATIQUES

Service de contrôle d'accès dans un environnement hospitalier
analyse des exigences et définition d'une solution adaptée pour les cliniques Saint-Luc

Coremans, Nascimo

Award date:
2011

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Faculté d'informatique
21, Rue Grandgagnage
5000 Namur

Année académique 2010-2011

Service de contrôle d'accès dans un environnement hospitalier

Analyse des exigences et définition d'une solution adaptée pour les Cliniques Saint-Luc

Nascimo Coremans

Mémoire présenté en vue de l'obtention du diplôme de Master en Sciences Informatiques

Remerciements

Je tiens à remercier particulièrement mon promoteur, Jean-Noël Colin, pour ses conseils et ses remarques qui ont été une aide précieuse pour écrire ce mémoire et également pour sa disponibilité et son soutient.

Je remercie également mon responsable de stage, Gery Mollers, et toute l'équipe du département informatique des Cliniques Saint-Luc pour leur accueil.

Un grand merci à ma famille et mes amis pour leur soutient et leurs encouragements. En particulier, je remercie ma maman et mon beau-père pour leurs relectures.

Résumé

La protection d'un système d'information en milieu hospitalier est importante afin de pouvoir assurer le bon fonctionnement de l'organisation et assurer une protection fiable et robuste des données personnelles. Cependant, le milieu hospitalier est bien souvent caractérisé par une grande hétérogénéité au niveau des métiers, du matériel et des logiciels rendant la gestion des utilisateurs et des accès complexe.

Ce mémoire tente de résoudre le problème du contrôle d'accès en proposant un service externalisé. Pour décrire ce service, il a été nécessaire de cerner les différentes exigences et fonctionnalités propres à cette organisation pour le contrôle de l'accès à ses applications. Ensuite, nous avons pu constituer l'état de l'art reprenant les principaux standards et protocoles actuels nécessaires pour proposer un service de contrôle d'accès.

Sur base cette base, il nous a été possible de définir une architecture détaillée des différents composants indispensables et de leurs interactions en se basant sur les standards SAML et XACML. Nous avons ensuite décrit les grandes étapes nécessaires au déploiement de la solution proposée. Enfin, nous avons analysé les menaces qui pesaient sur notre solution et la manière dont elle y répond.

Mots clés : hôpital, contrôle d'accès, hétérogénéité, SAML, XACML

Abstract

The protection of hospitals' information systems is important to ensure the correct operation of the organization and provide a reliable and robust protection of patient data. However, the hospital is often characterized by a great heterogeneity in the professionals, equipment and software that makes managing users and access complex.

This thesis attempts to solve the problem of access control by providing an outsourced service. To describe this service, it was necessary to identify the different requirements and features unique to this organization for access control to its applications. Then we were able to establish a state of the art including the main current standards and protocols that are required to provide an access control service.

From this point, it was possible to develop a detailed architecture of individual components required and their interactions based on SAML and XACML standards. We then described the major steps required to implement the proposed solution. Finally, we analyzed the threats to our solution and how it responds.

Keywords : hospital, acces control, heterogeneity, SAML, XACML

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte

Nous étudions le *Système d'Information* (SI) des Cliniques Universitaires Saint-Luc. On définit un SI par « un ensemble organisé de ressources (matériels, logiciels, personnel, données et procédures) qui permet de regrouper, de classifier, de traiter et de diffuser de l'information sur un environnement donné »[Wik]. Ce système d'information est spécifique puisqu'il concerne un hôpital et représente un outil de travail indispensable pour le personnel.

Dans le SI de Saint-Luc, nous avons pu constater quelques particularités propres à la plupart des systèmes d'information hospitalier :

- Il y a une grande quantité de personnel et celui-ci est constitué d'un nombre important de professions différentes : médicale, para-médicale, administrative, informaticiens, cuisiniers, ...
- Les différents groupes de métiers ont besoin d'applications spécifiques afin de les aider à réaliser leurs tâches quotidiennes. Il y a donc un nombre important d'applications d'un degré plus ou moins critique.
- Le parc informatique est très varié car il doit répondre aux besoins des différents services.
- Les données traitées par les différentes applications peuvent être sensibles(données personnelles ou médicales).

Le système d'information est donc une ressource essentielle et nécessaire pour les cliniques et son personnel. Il est dès lors important de le protéger. La présence de données sensibles oblige les cliniques à respecter un certain nombre de lois et règlements relatifs à la vie privée et aux données médicales. Les membres du personnel étant nombreux et ayant des caractéristiques très variées, il convient de n'autoriser que les actions légitimes sans pour autant les gêner dans leur travail. Enfin, la présence d'un matériel hétérogène doit pouvoir être pris en compte.

Dans ces conditions, il est nécessaire de pouvoir définir des règles claires sur la manière dont doit être protégé le système d'information. L'ensemble de ces

règles et procédures est ce qu'on appelle une politique de sécurité. Une politique de sécurité prend en compte 3 axes [Poi] :

- physique : pour protéger les locaux et les biens et pour contrôler l'accès physique au matériel informatique
- administratif : pour gérer le point de vue organisationnel de la sécurité comme la répartition des tâches , la séparation des pouvoirs, ...
- logique : pour contrôler l'accès aux applications et aux données

Nous ne nous intéresserons, dans ce mémoire, qu'à l'aspect logique de la politique de sécurité et plus précisément à la gestion des accès aux applications. La définition d'une politique est importante mais il s'agit ensuite d'en permettre la mise en œuvre et d'en vérifier son utilisation. La gestion du contrôle d'accès et leurs journalisations ne peut cependant pas être réalisée si une gestion des identités n'est pas mise en place. En effet, pour pouvoir autoriser un accès ou pour imputer une action à quelqu'un il est nécessaire de connaitre son identité. La gestion des identités, la gestion des accès et l'enregistrement des actions forment ensemble ce qu'on appelle l' « Identity and Access Management (IAM) ».

1.2 Système existant

Dans ce contexte, nous analyserons la manière dont le système actuel à Saint-Luc répond aux besoins de sécurité.

Le système, représenté sur la figure 1.1, est composé de plusieurs applications spécifiques aux différentes catégories de personnel, d'un composant assurant les fonctionnalités de sécurité (Huissier), d'une application permettant l'administration des utilisateurs, des équipements, des points d'accès et des politiques et la réalisation des audits (AdminSec) et enfin d'une base de données contenant toutes les informations nécessaires à l'Huissier pour fournir un service de sécurité.

1.2.1 Application

Il existe aussi différents types d'applications : des applications web, client/serveur ou 3-tiers(client/serveur d'application/serveur de données). Elles sont développées dans différents langages, même si l'accent a été mis sur le développement des applications Java ces dernières années. En fonction de leur architecture et/ou de leur langage de programmation, elles utiliseront l'Huissier de différentes manières. Nous allons voir dans la section suivante les services que l'Huissier fournit, comment il fonctionne et de quelles manières les différents types d'application font appel à ses services.

1.2.2 Huissier

Les applications devant être sécurisées utilisent un composant développé par Saint-Luc : l'Huissier. Ce composant remplit différentes fonctions qui sont :

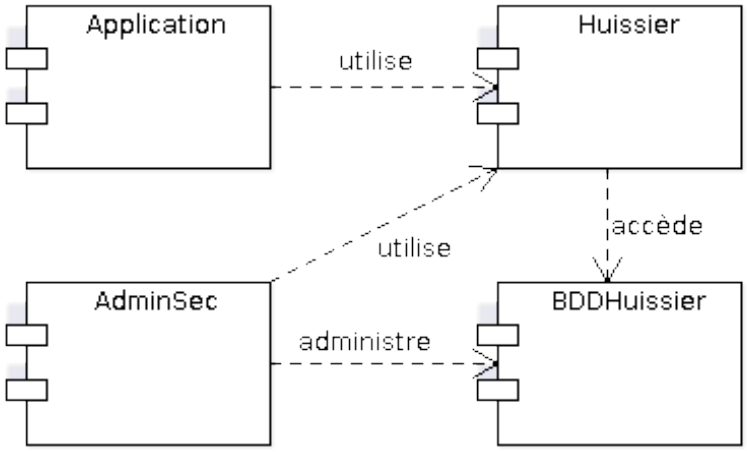

FIGURE 1.1 – Composants principaux

- dialogue de login
- authentification de l'utilisateur
- diffusion du profil de l'utilisateur
- évaluation des règles de sécurité
- journalisation
- détection du timeout
- diffusion du timeout

Ce module est implémenté de deux façons différentes. La première implémentation fonctionne comme une librairie accessible sous forme d'un serveur OLE Automation. La seconde implémentation est une interface Java qui est elle même implémentée en fonction de 3 modes d'utilisation.

Huissier sous forme de serveur OLE Automation

L'Huissier est implémenté sous la forme de serveur OLE Automation. OLE Automation[Mic] est un mécanisme de communication inter-processus qui permet à une application d'exposer des objets pour qu'ils puissent être manipulés par d'autres applications. L'application qui expose ses fonctionnalités est appelée serveur d'automation et celle qui les manipule, client d'automation. Les objets exposés par le serveur ont une interface externe bien définie qui permet aux clients d'utiliser ses propriétés et ses méthodes et ainsi utiliser les fonctionnalités proposées par le serveur. L'Huissier propose donc ses différentes fonctionnalités en tant que serveur d'automation et les applications jouent le rôle de client d'automation.

Sur la figure 1.2 est représentée la manière dont est déployé l'Huissier en tant que serveur OLE Automation. Lorsque l'application Huissier.exe est démarrée, elle charge la DLL Huissier.dll qui elle même charge la DLL HuiOLEAS. Lorsqu'une application est lancée, elle charge la DLL HuiOLEAC qui recherche une instance

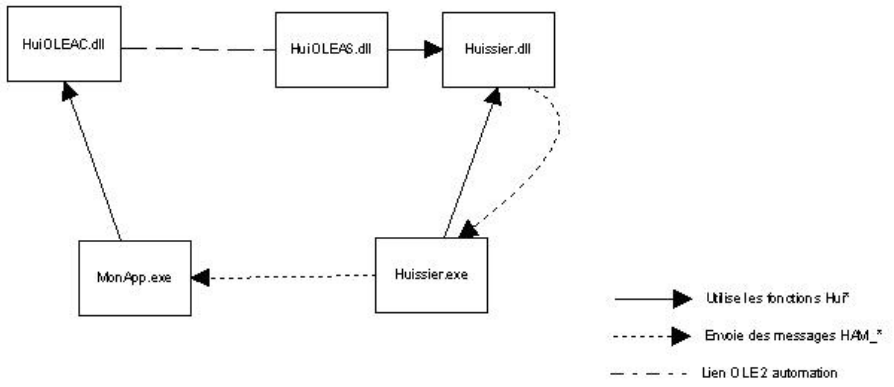

FIGURE 1.2 – Implémentation de l'Huissier via OLE Automation

d'objet d'automation Huissier et crée une interface vers cet objet. L'application peut ensuite utiliser les différentes fonctions de l'objet exposé via cette interface. La plupart des applications utilisent cette implémentation à l'exception des applications Java et web qui utilisent l'Huissier via une interface Java que nous décrivons dans la section suivante.

Huissier sous forme d'interface Java

Afin que les applications web et Java puissent utiliser les services de l'Huissier, l'implémentation de l'Huissier sous forme d'une interface Java a été nécessaire.

Une application désirant utiliser les services de l'Huissier doit implémenter l'interface Application ou WebApplication. Cette interface est constituée de quelques fonctions nécessaires à l'Huissier afin d'obtenir des informations sur l'application et pour que celle-ci puisse intercepter les différents messages qu'elle reçoit de l'Huissier. L'application va récupérer un objet UserManager. Ce UserManager est une interface qui peut être implémentée de trois manières différentes en fonction du mode d'utilisation utilisé. Les modes existants sont les modes natif, serveur ou web.

Le mode natif utilise l'Huissier sous la forme de serveur OLE Automation vu dans la section précédente. Grâce à une passerelle JAVA/COM, il est possible pour les applications Java de communiquer avec le serveur d'automation de l'Huissier. Seul l'Huissier exécuté sur le poste de travail est utilisé.

Le mode serveur est utilisé lorsque le mode natif ne peut pas l'être, par exemple parce qu'il n'y a pas d'Huissier présent sur le poste de travail de l'utilisateur. L'application utilise alors une implémentation Java de l'Huissier qui ré-implémente la plupart des fonctions de l'Huissier natif. Cette implémentation est déployée sur un serveur d'application JBoss et accessible en tant qu'EJB par les applications. Cet EJB est sécurisé et demande au client de s'authentifier. Le module d'authentification JBoss est remplacé par un HuiLoginModule qui per-

mettra d'authentifier l'utilisateur. Il existe cependant quelques différences entre l'implémentation Java et l'Huissier natif. La détection de l'activité, par exemple, se fait pour l'application et non pour le poste, le serveur n'ayant pas accès à l'activité du poste de travail.

Le mode web est, quant à lui, destiné aux applications web. Il utilise également l'implémentation Java déployée sur le serveur d'application JBoss. Les fonctions sont accessibles via une servlet et pas via des EJB comme pour le mode serveur.

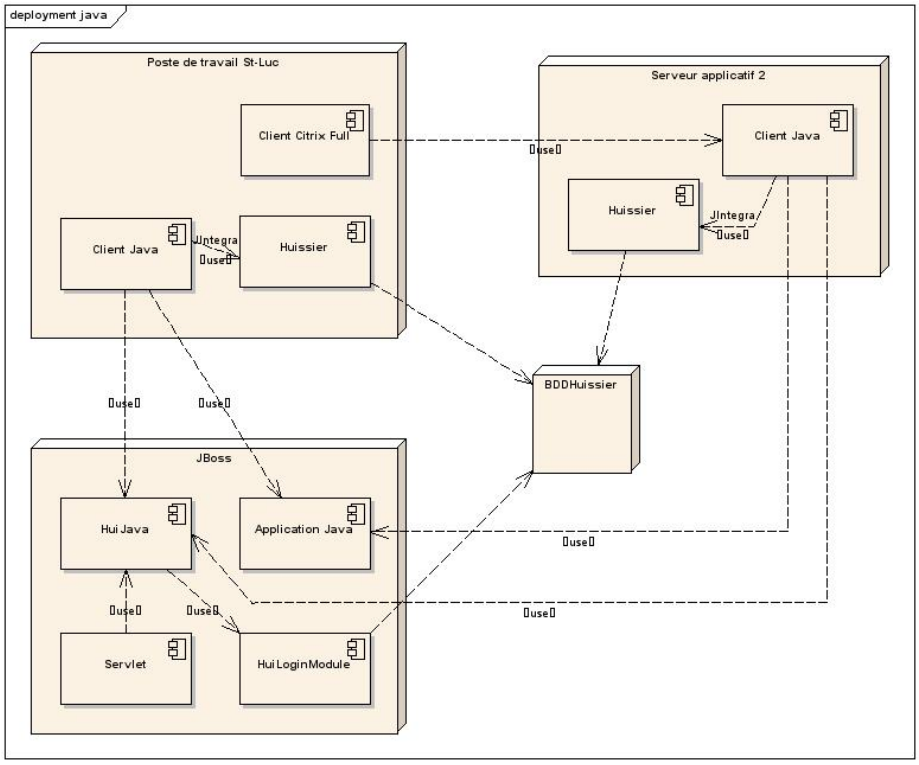

FIGURE 1.3 – Architecture de l'Huissier

XenApp de Citrix

Dans certaines situations, les application sont virtualisées via XenApp comme on peut le voir sur le schéma ci-dessus 1.3. L'architecture XenApp est composée d'une ou plusieurs fermes de serveurs applicatifs. Chaque ferme contient plusieurs serveurs affectés à un groupe d'applications. Ils sont alors responsables de la disponibilité de l'application. Pour chaque ferme, un serveur est désigné comme data collector qui tiendra en quelque sorte le rôle de chef d'orchestre. A l'aide des différentes informations qu'il reçoit des autres serveurs, il peut orienter un utilisateur vers le serveur adéquat en fonction de l'application qu'il demande et de l'état des serveurs. Enfin, un serveur hors de la ferme sert de passerelle pour l'utilisateur. Celui-ci y accède soit directement via un client Xe-

nApp installé sur son poste de travail, soit via son navigateur en passant par une interface web. Cette passerelle lui permet de s'authentifier auprès du système XenApp et de récupérer la liste des applications auxquelles il a accès. Il ne lui reste plus ensuite qu'à demander une application qui sera lancée pour lui sur un des serveurs applicatifs.

L'utilisation de XenApp permet de fournir plus facilement un accès aux applications, de les modifier plus facilement et surtout, soustrait le service informatique de l'installation et de la configurations des applications pour chaque poste, seule l'installation du client XenApp étant nécessaire.

Cependant, l'utilisation de XenApp et de l'Huissier pose quand même quelques problèmes. En effet, chaque application devant être sécurisée utilise les services de l'Huissier. Des applications appartenant à des groupes différents sont exécutées sur des serveurs différents. Un Huissier est donc lancé sur chaque serveur. En conséquence, plusieurs Huissiers peuvent être exécutés en même temps pour un seul utilisateur. Ceci implique que chaque Huissier ouvert possède son propre time-out. Chaque Huissier affiche donc sa fenêtre de login et empêche l'utilisateur de continuer son activité alors qu'il est actif.

L'utilisation de XenApp a donc induit une obligation pour l'utilisateur de s'authentifier autant de fois qu'il y a de Huissiers exécutés. Chaque Huissier ayant un time-out propre, l'utilisateur est confronté plusieurs fois à l'apparition d'un écran noir et d'une demande de login alors qu'il est toujours actif. Afin de remédier à ces problèmes, un HuissierService a été développé afin de synchroniser le time-out des différents Huissiers et de masquer les différentes demandes de login et écrans noirs qui surviennent alors que l'utilisateur est actif. Ce HuissierService doit être lancé sur le poste de travail exécutant le client XenApp et communique avec les différents Huissiers démarrés.

1.2.3 Administrateur de sécurité

L'administrateur de sécurité est une application offrant différentes fonctionnalités. Ces fonctionnalités sont :

- la gestion des utilisateurs et de leurs profils
- la gestion des accès
- la gestion des traitements
- la gestion des points d'accès
- la surveillance du système (audit)

1.2.4 Base de données Huissier

La base de données Huissier contient toutes les informations nécessaires à l'Huissier et à l'Administrateur de sécurité. Elle comprend les données sur les utilisateurs, les profils, les droits d'accès et les points d'accès mais également l'enregistrement des sessions et des accès des utilisateurs afin de pouvoir surveiller le système.

1.3 Problématique

Le système existant arrive à ses limites et différents problèmes apparaissent.

Tout d'abord, le système informatique présent est très hétérogène, les systèmes d'exploitation sont multiples (Unix, Windows Server)et différents systèmes de gestion de base de données cohabitent (SQLServer, Oracle, Sybase, ...). L'architecture des applications peut être de type client-serveur , de type 3 tiers ou de type web aussi bien intranet que extranet. La communication entre ces différentes applications est de plus en plus difficile à mesure que de nouvelles technologies apparaissent, celles-ci devant pouvoir s'intégrer au système existant.

Ensuite, on peut ajouter à cela un besoin de plus en plus important d'avoir une démarche certificative dans la gestion des accès aux applications et de la protection des données en général. Aujourd'hui, le responsable de la sécurité du système informatique doit pouvoir être en mesure de montrer que la politique de sécurité est bien mise en œuvre mais également qu'elle est respectée ou au contraire violée.

Et enfin, le système actuel devant intégrer un nombre important d'applications et de technologies différentes, il est de plus en plus difficile de proposer aux utilisateurs une procédure de connexion simple tout en garantissant une bonne sécurité. Pourtant, le milieu médical requiert des procédures souples et les moins contraignantes possibles afin de ne pas gêner le personnel dans son travail.

Le but de ce mémoire sera donc de trouver un moyen d'assurer le contrôle d'accès logique dans un environnement hétérogène, en se plaçant dans une démarche certificative et en offrant des procédures simples pour l'utilisateur. L'utilisation de XenApp sera écartée du problème car il est difficile d'obtenir des informations précises sur son fonctionnement et son architecture.

1.4 Méthodologie

Afin de répondre à notre problématique, nous appliquerons la démarche suivante :

1. Sur base du cas de Saint-Luc, nous établirons les exigences propres à une solution de contrôle d'accès
2. Nous détaillerons les protocoles et standards existants les plus connus dans le domaine de l'authentification et de l'autorisation.
3. Nous examinerons l'architecture de quelques solutions existantes.
4. Nous proposerons une solution répondant à notre problématique et ses exigences sur base des standards et protocoles existants.
5. Nous analyserons les différentes menaces et faiblesses de notre solution en matière de sécurité.

Chapitre 2

Fondements et concepts de sécurité informatique

Sur base de la littérature existante, nous pouvons définir les différents fondements et concepts de sécurité informatique. Nous commencerons par expliquer ce qu'est le chiffrement. Nous continuerons par l'explication des objectifs de sécurité. Ces objectifs pourront être réalisés grâce à différentes fonctions que nous expliciterons également. Nous aborderons ensuite quelques préoccupations qui ne concernent pas directement la sécurité mais qu'il nous semble important de prendre en compte lors de l'élaboration ou le choix d'un système informatique. Enfin, nous terminerons par décrire ce qu'est une politique de sécurité.

2.1 Chiffrement

Le chiffrement consiste à « convertir des données lisibles de manière à les rendre illisibles sans connaissance supplémentaire »[Col]. Il se base sur des algorithmes cryptographiques utilisant un secret qu'on appelle une clef. Il y a deux types d'algorithmes : les algorithmes symétriques et les algorithmes asymétriques.

2.1.1 Cryptographie symétrique

La cryptographie symétrique se base sur une clef secrète. Cette clef unique est utilisée pour chiffrer et déchiffrer les données. Elle est donc partagée par les deux entités désirant communiquer de manière sécurisée. Cette clef doit donc rester secrète, ce qui suppose qu'elle soit distribuée ou échangée de manière sûre afin que personne d'autre n'en prenne connaissance.

2.1.2 Cryptographie asymétrique

La cryptographie symétrique repose sur une paire de clef, une clef privée et une clef publique. Ces deux clefs sont distinctes et ne peuvent se déduire l'une de

l'autre. La clef publique permet le chiffrement et la clef privée le déchiffrement. Comme son nom l'indique, la clef publique est accessible à tous et la protection de la clef privée est à charge de l'utilisateur. Cette méthode simplifie donc la gestion des clefs mais est en revanche beaucoup plus lente que le chiffrement par clef symétrique et ne permet pas de chiffrer de gros volumes de données.

Certificat numérique et infrastructure à clef publique

L'utilisation de clefs publiques pose également un autre problème. Comment être sûr qu'une clef publique appartient bien à l'entité avec laquelle nous voulons communiquer de manière sécurisée ? Il est donc nécessaire de pouvoir lier une clef publique à un utilisateur. Ce lien est ce qu'on appelle un certificat numérique. Un tiers de confiance doit donc se porter garant de ce lien.

La mise en place d'une infrastructure à clefs publiques (Public Key Infrastructure, PKI) permet de protéger l'intégrité et l'authenticité des clefs[Col]. De cette manière, nous pouvons être sûrs que le certificat n'est pas modifié et qu'il appartient bien à son possesseur. Une PKI est contient les composants suivants :

Autorité de certification Cette entité gère, émet et révoque les certificats. Chaque certificat est signé avec sa clef. Sa clef est auto-signée.

Autorité d'enregistrement Cette entité vérifie l'identité des nouveaux utilisateurs et les aide dans la gestion de leur clef.

Autorité de dépôt Cette entité stocke les certificats.

Autorité de recouvrement Cette entité stocke les clefs privées générées de manière sécurisée et permet de les restaurer suite à la perte de la clef ou à une contrainte légale.

2.2 Objectifs de sécurité

2.2.1 Disponibilité

La disponibilité est définie comme la « propriété d'être accessible et utilisable à la demande d'une entité autorisée »[ISO09]. Le but est de pouvoir garantir à l'utilisateur autorisé que le système est disponible. La disponibilité demande le plus souvent une grande fiabilité des composants du système. En effet, si un composant est inaccessible, il devient difficile de garantir la disponibilité du système entier. Cependant, la disponibilité n'est pas qu'un problème de fiabilité. Elle peut être garantie par d'autres moyens. Le déni de service est typiquement une attaque qui affecte la disponibilité du système.

2.2.2 Confidentialité

La confidentialité est caractérisée par le fait que « l'information n'est pas rendue accessible ou divulguée à des individus, entités ou processus non-autorisés »[ISO09]. Le but est de protéger l'information des divulgations non-autorisées. Différentes attaques peuvent menacer la confidentialité, que ce soit sur des données stockées

ou lors de l'échange ou du traitement de ces données. La garantie de la confidentialité nécessite le plus souvent le recours à des mécanismes de chiffrement.

2.2.3 Intégrité

L'intégrité est définie comme la « propriété de protéger l'exactitude et la complétude des actifs »[ISO09]. Le but est de garantir que les données ne sont pas altérées ou détruites d'une manière non-autorisée. Cette propriété s'applique aux actifs stockés, échangés ou en cours de traitement. Pour les données stockées, l'intégrité peut en grande partie être garantie en appliquant un contrôle d'accès. Lorsque les données sont échangées, l'accès ne peut plus être contrôlé. La solution est alors de signer les données de manière électronique.

2.3 Fonctions de sécurité

2.3.1 Authentification

Selon la RFC 2828[Shi00], l'authentification est la procédure qui permet de vérifier l'identité revendiquée par une entité. Le but est de s'assurer que la personne ou le service est bien celui qu'il prétend être. L'authentification se déroule en deux étapes :

- l'identification qui consiste à présenter un identifiant
- la vérification qui consiste à fournir ou générer une information d'authentification qui permet de lier l'identifiant et l'entité.

Le processus d'authentification est très important car l'authentification des entités est la base d'autres services de sécurité comme le contrôle d'accès ou l'auditabilité. Il se base sur l'échange d'informations pour vérifier l'identité d'une entité. Cette information est ou est dérivée de :

- quelque chose que l'entité connait (mot de passe, code d'identification, ...)
- quelque chose que l'entité possède (jeton, carte à puce, certificat électronique, ...)
- quelque chose que l'entité est (caractéristiques physiques)
- quelque chose que l'entité sait faire (signature manuscrite, reconnaissance vocale)

Authentification unique

L'authentification unique ou Single Sign-On(SSO) consiste à ne demander qu'une seule authentification pour un ensemble de services, d'applications et/ou de sites internet. La mise en place d'un système de Single Sign-On apporte différents avantages décrits entre autres dans [Sal03].

Tout d'abord, un système de SSO apporte un avantage ergonomique pour l'utilisateur. En effet, l'utilisateur ne doit s'identifier qu'une seule fois. Les procédures d'authentification sont ainsi simplifiées. Le profil de l'utilisateur est

propagé aux différentes applications sans qu'il ne s'en rende compte.

Ensuite, la sécurité du système se voit améliorée. Le serveur d'authentification étant le seul à recueillir des informations de l'utilisateur, il est plus facile de concentrer son attention sur la sécurisation de ce serveur. La mise en place d'une politique de sécurité cohérente pour l'ensemble des applications est également plus facile. Enfin, un service commun d'authentification sera beaucoup plus facile à faire évoluer qu'un ensemble de services hétérogènes spécifique à chaque application.

Enfin, l'utilisation d'un service commun d'authentification permet de développer de nouvelles applications en délégant le service et non plus en redéveloppant un nouveau service d'authentification à chaque fois. De plus, l'ensemble des applications bénéficie des mêmes garanties de sécurité. La mise en place d'un SSO nécessite un référentiel commun d'utilisateurs, la gestion des comptes en est donc simplifiée et une politique cohérente de gestion des comptes peut facilement être mise en place.

2.3.2 Autorisation

« L'autorisation est un droit ou une permission qui est accordé à une entité pour accéder à une ressource. »[Shi00] Le processus d'autorisation consiste à décider d'octroyer ou non ces droits à une entité. Afin de prendre cette décision, il est nécessaire d'avoir authentifié auparavant l'entité. La décision pourra alors se baser sur un ensemble de politiques de droit d'accès mais aussi sur les attributs de l'entité, celle-ci ayant été authentifiée.

2.3.3 Auditabilité

L'auditabilité consiste à vérifier que les différents systèmes de contrôle, les différentes procédures et politiques de sécurité sont bien respectées et de détecter les failles de sécurité. Cette fonction sera réalisée grâce aux fonctions de traçabilité et de non-répudiation.

Traçabilité

La traçabilité consiste à enregistrer les actions effectuées par une entité. Il faut donc enregistrer aussi bien les accès que les tentatives d'accès à un service de la part d'une entité. Ces enregistrements devront pouvoir être conservés et si nécessaire pourront être exploités.

Non-répudiation

La non-répudiation consiste à établir un lien irréfutable entre une action et l'entité qui en est à l'origine. De cette manière, l'entité ne pourra pas nier être à l'origine d'une action.

2.3.4 Contrôle d'accès

Le contrôle d'accès consiste à protéger les ressources du système contre un accès non-autorisé[Shi00] . Pour ce faire, il faut vérifier l'identité de l'entité (authentification) et vérifier si elle possède bien les droits nécessaires (autorisation). Dans certains cas, il peut être utile d'utiliser la traçabilité afin de vérifier certaines conditions indispensables à l'obtention du droit. La traçabilité permet également de responsabiliser les entités, chacune de leurs actions étant enregistrées.

Il existe différents types de contrôle d'accès :

Mandatory Access Control(MAC) La décision de contrôle repose sur des permissions définies à priori et sur lesquelles l'utilisateur n'a aucun contrôle.

Discretionary Access Control(DAC) La décision se base sur les permissions accordées par le propriétaire de la ressource.

Role-Based Access Control(RBAC) La décision de contrôle d'accès est basée sur le rôle de l'utilisateur. Le rôle découle de la structure de l'organisation et correspond généralement à une fonction au sein de l'organisation.

2.4 Préoccupations

2.4.1 Interopérabilité

≪ En informatique, l'interopérabilité peut se définir comme la capacité, pour deux ou plusieurs systèmes informatiques, à fonctionner ensemble. ≫[Inta] La communication entre les systèmes informatiques doit donc répondre à des normes afin qu'ils puissent communiquer sans difficultés. Ces normes décrivent des exigences sur les formats de données échangées et la manière de les échanger.

2.4.2 Utilisabilité

L'utilisabilité est selon la norme ISO 9241-11 définie comme ≪ le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ≫. L'utilisabilité est importante dans le domaine de la sécurité informatique. Les utilisateurs manipulent des données sensibles, il est donc nécessaire qu'ils effectuent les différentes actions qui leur sont demandées de manière efficace et efficiente.

2.4.3 Confiance

La confiance est une notion difficile à définir. Nous utiliserons une des définitions proposées dans [AJ], à savoir la ≪ confiance de décision ≫. Cette définition peut être traduite comme ceci : ≪ La confiance est la mesure dans laquelle une partie est disposée à dépendre de quelque chose ou de quelqu'un dans une situation donnée avec un sentiment de relative sécurité ≫. En sécurité informatique, il est important que les entités communicantes puissent établir une relation de

confiance entre elles puisqu'elle dépendent les unes des autres. Les fonctions de contrôle d'accès, d'authentification et d'autorisation reposent généralement sur la confiance entre les entités ou entre les entités et une autorité.

2.5 Politique de sécurité

La RFC 2828[Shi00] définit la politique de sécurité comme un ensemble de règles et pratiques qui spécifie ou réglemente comment un système ou une organisation fournit des services de sécurité pour protéger les ressources sensibles et critiques du système. Une politique de sécurité peut se développer dans trois directions distinctes : les politiques de sécurité physique, administrative et logique[Poi]. La politique de sécurité physique est destinée à protéger les ressources matérielles en y contrôlant l'accès.

La politique de sécurité administrative traite du point de vue organisationnel de la sécurité. Les règles portent sur la répartition des tâches au sein de l'entreprise afin d'éviter des dysfonctionnements liés à un trop grand pouvoir de certaines personnes au sein de l'entreprise.

La politique de sécurité logique concerne l'accès aux ressources logiques de l'entreprise. La gestion du contrôle d'accès logique repose généralement sur trois services : identification, authentification et autorisation.

L'ITSEC a défini une série de critères qui définissent les fonctions qui doivent être mises en œuvre afin de protéger le système d'information[CCC]. Elle définit également les preuves qui doivent être apportées afin que les critères sont bien respectés. Ces critères sont une base pour la définition d'une politique de sécurité complète..

Chapitre 3

Fonctionnalités et exigences

3.1 Fonctionnalités

3.2 Exigences de l'environnement

Dans cette section, nous identifions les exigences d'une organisation pour le contrôle d'accès logique aux applications. Pour cela, nous nous basons sur les exigences des cliniques Saint-Luc en essayant cependant de rester général afin de pouvoir correspondre à celles rencontrées par la plupart des organisations.

Nous commencerons par identifier les différentes parties prenantes du système et les buts généraux qu'elles poursuivent. Nous détaillerons ensuite ces buts jusqu'à obtenir des critères ou fonctionnalités.

3.2.1 Parties prenantes

Tout d'abord, il nous semble important de définir les différents acteurs prenant part au système. Ces acteurs sont :

- les utilisateurs des différentes applications du système informatique
- les développeurs d'applications internes
- les développeurs d'applications externes
- les administrateurs du système informatique
- le responsable de la sécurité informatique
- les hackers ou personnes mal intentionnées
- le helpdesk/support aux utilisateurs
- les organes régulateurs

3.2.2 Buts

Chaque partie prenante possède un ou plusieurs buts qu'elle espère réaliser.

Parties prenantes	Buts
Utilisateur	Accéder facilement aux applications sécurisées
Développeur interne ou externe	Protéger son application des accès non autorisés
Administrateur du système	Gérer les utilisateurs, les profils, les points d'accès et l'équipement du système d'information.
Responsable de la sécurité	Contrôler la cohérence et l'intégrité du système d'information
Hacker	Accéder à des informations ou ressources non autorisées
Helpdesk	Aider les utilisateurs à utiliser le système et les développeurs à le maintenir
Organe régulateur	Etablir des lois et vérifier leur application

3.2.3 Fonctionnalités

Les fonctionnalités recherchées par Saint-Luc sont les suivantes :
- l'authentification
- l'autorisation
- le contrôle d'accès
- l'audit
- l'administration

3.2.4 Exigences

Suivant les différents buts des parties prenantes et des fonctionnalités attendues, un certain nombre d'exigences s'imposent à l'éventuelle solution.

Contrôle d'accès

La fonctionnalité première est le contrôle d'accès. Le contrôle d'accès du système actuel est souvent lié au contexte local de l'application. Les droits accordés portent généralement sur l'application ou sur un module de l'application. Dans certains cas, l'application possède ses services de sécurité propres et sa base d'utilisateurs. Dans les autres cas, l'application utilise les services de l'Huissier. Il est donc nécessaire, afin de garantir une cohérence pour le système d'information, de proposer un service de contrôle d'accès externe et unique qui serait utilisé par toutes les applications.

De plus, la multiplication des bases utilisateurs peut amener une multiplication des comptes pour une seule et même personne. La gestion des bases d'utilisateurs devient donc complexe et des incohérences peuvent apparaitre. Il est donc nécessaire d'unifier toutes ces bases utilisateurs en un référentiel central. Dans une institution hospitalière, comme c'est le cas à Saint-Luc, il y a généralement un nombre important d'utilisateurs ayant des fonctions différentes.

Le contrôle d'accès est donc basé sur les rôles, les droits liés au métier. En plus de cela, il est souvent nécessaire de pouvoir attribuer des droits complémentaires et spécifiques dont l'utilisateur à besoin.

Un service de contrôle d'accès complet nécessite un service d'authentification, d'autorisation et d'audit. Nous allons voir à présent les exigences qui portent sur ces fonctionnalités.

Authentification

L'authentification des utilisateurs se fait actuellement à l'aide d'un badge, d'un couple identifiant/mot de passe ou un code pin. L'Huissier n'arrive plus à intégrer toutes les applications et technologies différentes. Cependant, il est important pour les utilisateurs d'avoir une procédure simple pour s'authentifier et ainsi ne pas être retarder dans leur travail. Dans ce sens, il faut pouvoir également proposer un service de Single Logout. Celui-ci pourra être inité par le retrait du badge ou à la demande explicite de l'utilisateur. Il pourra également être initié lorsque l'utilisateur est inactif durant un certain moment. Une gestion de session et la gestion d'un time-out devront être mis en place.

L'authentification simple, donc ne reposant que sur un seul facteur, peut convenir dans la plupart des situations. Cependant, pour des événements sensibles, il est nécessaire de proposer une authentification forte, basée sur 2 facteurs, et garantissant la non-répudiation des événements effectués. Il existe trois grandes technologies d'authentification forte : le mot de passe à usage unique (One Time Password ou OTP), les certificats numériques et la biométrie. Dans notre cas, le support devra être une carte à puce facile à associer au badge existant. Le plus simple est donc d'utiliser la combinaison d'un certificat PKI détenu sur une carte à puce et d'un mot de passe. De plus, le certificat pourra offrir des services de signature électronique et de chiffrement.

Il est évident que ce n'est pas aux applications de gérer les moyens par lesquels l'utilisateur s'authentifie. Il est même préférable que les applications n'aient pas connaissance de ces informations. Une autorité en qui les applications ont confiance devra donc leur affirmer que l'utilisateur est bien authentifié. La confiance entre les applications et l'autorité se fera également à l'aide dans certificat. Une infrastructure de gestion de la confiance devra donc être mise en place.

Autorisation

Les autorisations seront obtenues suite à la demande d'une décision d'autorisation. Cette décision sera prise sur base de l'évaluation de la politique de sécurité applicable, de l'utilisateur(rôle, attributs, droits) et du contexte dans lequel l'autorisation désire être obtenue(lieu ou temps). Les informations nécessaires à la décision doivent provenir d'un référentiel central. La politique doit être unique et centralisée dans un référentiel. Le profil de l'utilisateur doit

également être centralisé dans un seul référentiel.

Audit

Le service d'audit doit permettre de tracer tous les événements ayant lieu dans un scénario de contrôle d'accès. La règle des 4W s'applique (Who, When, What, Where). Il est donc nécessaire d'enregistrer pour chaque événement, l'auteur qui en est à l'origine, le moment où il s'est produit, la nature de l'événement et l'endroit où il s'est produit. L'endroit où il s'est produit sera ici l'application ou le module de l'application. Toutes ces traces vont permettre au responsable de la sécurité de vérifier que la politique de sécurité est bien appliquée et ainsi être en mesure de produire des rapports de conformité ou dans le cas contraire de détecter les violations et de pouvoir y remédier.

Administration

L'administration des éléments nécessaires à un service de contrôle d'accès est importante. Elle doit permettre de gérer les utilisateurs, les équipements et points d'accès, les politiques de sécurité et les groupes ou rôles.

La gestion des utilisateurs consiste à pouvoir gérer le cycle de vie de l'utilisateur dans le système d'information (création, modification, suppression du profil) mais aussi de pouvoir suspendre son compte. Il faut également pouvoir gérer ses moyens d'authentification (mot de passe, certificat).

L'important est de pouvoir proposer une interface unique d'administration basée sur un référentiel commun afin d'avoir une gestion des identités et des accès cohérente pour l'ensemble du système d'information. Le système existant possédant déjà des annuaires ou référentiels, il sera sans doute nécessaire de les synchroniser et de les utiliser pour approvisionner le référentiel commun.

Sécurité

Certaines exigences sont propres à la sécurité en général. Le contrôle d'accès permet d'assurer une intégrité et une confidentialité en protégeant l'accès aux ressources accessibles via les différentes applications. Afin d'être sûr de cela, nous étudierons plus en détail les considérations à prendre en compte afin de garantir une bonne sécurité une fois la solution proposée.

3.3 Architecture des applications

Le problème d'hétérogénéité du système est en partie dû aux différents types d'applications. Chaque application nécessitera donc des choix différents dans la construction de la solution. Nous définissons ci-dessous les types d'applications présents dans le système de Saint-Luc afin de présenter une architecture générique qui pourra être utilisée en vue d'y appliquer une solution.

3.3.1 Application 2-tier ou client/serveur

Dans cette architecture, le client demande une ressource et le serveur lui fournit directement. Ces requêtes peuvent aussi bien concerner une page web, un fichier ou une information contenue dans une base de données. Si tous les traitements sont effectués sur le serveur, on parlera d'un client léger dont la seule charge sera la présentation des données. C'est le cas typique du navigateur web. Si en revanche, une partie, voire tous les traitements sont effectués sur le poste client, on parlera d'un client lourd.

3.3.2 Application 3-tier

L'architecture 3-tiers est partagée en 3 niveaux : le client, le serveur d'application et le serveur de données. Le client demande toujours des ressources comme dans l'architecture 2-tiers. Le serveur d'application est chargé de fournir les ressources demandées mais pour cela il fait appel à un autre serveur, le serveur de données. Le serveur de données fournit les données dont le serveur d'application à besoin. Le client se charge donc de la présentation, le serveur d'application de la partie métier et le serveur de données comme son nom l'indique, des données.

3.3.3 Application web

Une application web n'est accessible que via un navigateur web, donc un client léger. L'application peut soit être basée sur une architecture 2-tiers, soit sur une architecture 3-tiers. Lorsque des capacités de traitements sont nécessaires, la partie métier de l'application est bien souvent hébergée sur un serveur d'application mais est accesible via un servlet hébergé sur un serveur web. L'architecture est alors constituée d'un navigateur web, d'un serveur web chargé de générer les pages web, du serveur d'application effectuant les traitements et du serveur de données.

3.3.4 Architecture générique du système

Nous supposerons donc que notre solution doit pouvoir s'appliquer à un environnement comprenant des applications 2-tiers, 3-tiers et web comme l'illustre le schéma ci-dessous.

Les applications du système peuvent donc être constituées de :

- un client lourd et un serveur de données
- un client léger, un serveur d'application et un serveur de données
- un navigateur, un serveur web et un serveur de données
- un navigateur, un serveur web, un serveur applicatif et un serveur de données

La solution proposée devra donc pouvoir s'adapter à ces différentes architectures.

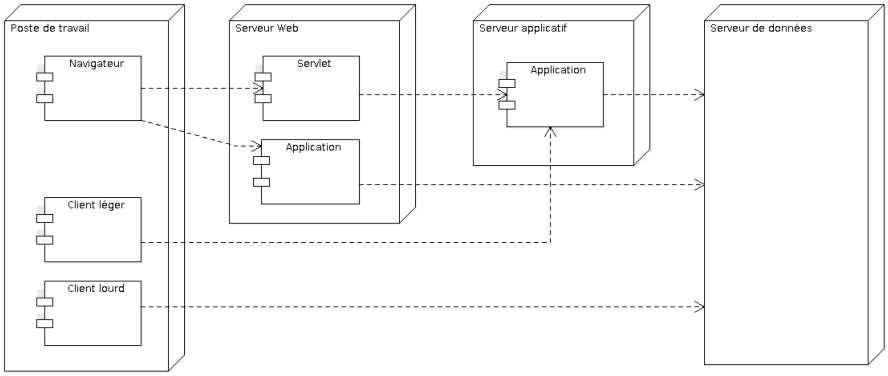

FIGURE 3.1 – Architecture générique

3.4 Récapitulatif des fonctionnalités et des exigences

Dans cette section, nous résumons les exigences nécessaires pour définir notre solution. Elles nous permettront d'analyser les différents standards existants et surtout de construire une architecture adaptée.

Fonctionnalité	Exigences
Contôle d'accès	– Service externe – Service unique – Utilisable par toutes les applications (interopérabilité) – Référentiel central pour la décision du contrôle d'accès – Basé sur les rôles
Authentification	– Utilisation des moyens d'authentification déjà présents (badge, identifiant/mot de passe, code pin) – Authentification unique (SSO) – Déconnexion unique (SLO) – Gestion de session (applications démarrées, inactivité) – Authentification forte et non-répudiation pour les actions sensibles – Signature et chiffrement – Gestion de la confiance
Autorisation	– Référentiel central pour la prise de décision – Basé sur le profil, la politique de sécurité et le contexte – Politique de sécurité unique
Audit	– Evènements tracés – Date et heure enregistrées – Source de l'évènement (application, module, ...) – Nature de l'évènement (demande d'accès, authentification, demande d'autorisation, ...) – Auteur à l'origine de l'évènement
Administration	– Interface d'administration unique (gestion centralisée) – Gestion des utilisateurs (utilisateurs, rôles, profils) – Gestion des équipements et points d'accès – Gestion des droits – Gestion de la politique de sécurité – Visualisation de l'état du système (traces, attributions des droits, ...) – Synchronisation et approvisionnement du référentiel central
Architecture	– Service de contrôle d'accès unique et externe – Référentiel centralisé pour les décisions de contrôle d'accès(utilisateurs et politiques) – Intégration des applications existantes (web, 2-tiers, 3-tiers)

Chapitre 4

Etat de l'art

Nous commencerons par présenter les service d'authentification que sont Kerberos, SAML et OpenID. SAML et OpenID permettent également le partage d'attributs. Nous présenterons ensuite XACML et OAuthSAML, deux protocole d'autorisation. XACML permet également d'exprimer une politique de sécurité. Nous verrons les possibilités qu'offre l'utilisation conjointe de SAML et XACML. Enfin, nous terminerons par étudier l'architecture des différentes solutions que sont Shibboleth, CAS et LemonLDAP.

4.1 Kerberos

Kerebros[Na05] est un service d'authentification distribué qui permet de vérifier l'identité d'une entité sur un réseau ouvert. Il ne s'appuie pas sur des assertions et suppose que des messages peuvent être lus, modifiés ou insérés. Kerberos utilise pour cela une séries de messages cryptés basé sur un secret partagé. Il est basé sur le protocole d'authentification Needham and Schroeder et apporte en plus un service de tickets.

Le protocole d'authentification Needham and Schroeder permet à des tiers, à l'aide d'une secret partagé, de se prouver l'un à l'autre leur identité. Avec l'aide d'un tiers de confiance, il leur est possible de partagé une clef de session uniquement connue par eux deux. Chaque tiers partage une clef avec le tiers de confiance qui est utilisée afin de crypté la clef de session. Chaque tiers est donc assuré que cette clef provient du tiers de confiance et que seul la clef de session n'est connue que d'eux et du tiers de confiance. Le protocole Kerberos ajoute cependant l'utilisation d'un nombre à usage unique (nonce) afin de se prémunir d'une attaque par rejeu.

Kerberos propose deux services, un service d'authentification et un service de tickets.

Le service d'authentification se déroule comme ceci : Un client envoie une requête au serveur d'authentification pour obtenir un certificat pour un serveur donné. Ce certificat est chiffré avec la clef du client et contient un ticket et une clef de session. Ce ticket contient l'identité du client et la clef de session chiffrée

avec la clef du serveur. En transmettant ce ticket au serveur, le serveur peut l'authentifier et ils partagent tout les deux une clef de session. A l'aide de cette clef, ils peuvent maintenant crypter leurs communications.

Lorsque l'utilisateur désire accéder à plusieurs services, il doit à chaque fois se re-authentifier auprès du serveur d'authentification. Le service de ticket permet d'éviter cela. Le client demande un ticket pour communiquer avec le service de tickets. Ce ticket (ticket-granting-ticket ou TGT) permettra au client d'obtenir d'autre tickets auprès du service de tickets seulement en présentant le TGT.

Au final, le protocole Kerberos suppose la présence de 4 entités : le client, le serveur, le service de tickets et le service d'authentification. Pour accéder aux services d'un serveur, le client doit pouvoir prouver son identité. Il va donc commencer par s'authentifier auprès du serveur d'authentification. Il reçoit en retour un ticket et une clef de session. Ce ticket contient des informations sur le client et la clef de session. Grâce à ce ticket, le client va contacter le service d'émission de tickets afin d'obtenir un ticket d'accès au serveur. Après vérification de l'authenticité de la requête, le service de tickets va délivrer au client un ticket d'accès au serveur. Le client va ensuite le fournir au serveur qui, après vérification, lui autorisera l'accès à ses services.

4.2 SAML

Security Assertion Markup Language (SAML) est un ensemble de spécifications techniques basées sur XML[MM05]. Ces spécifications visent à fixer un cadre pour la communication des informations de sécurité. La conception du standard a été faite de manière a ce qu'elle puisse être customisée si nécessaire et ensuite adoptée par d'autres standard. Il est ainsi utilisée comme base technique pour d'autres standard tel que Liberty Alliance, Shibboleth ou WS-Security.

SAML est défini en terme d'assertions, protocoles, bindings et profiles comme le montre le schéma ci-dessous[]. Nous allons voir plus en détail de quoi il s'agit.

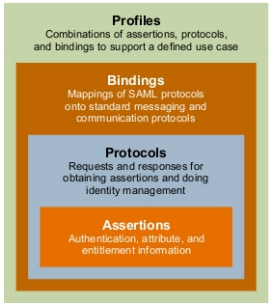

FIGURE 4.1 – Concepts SAML

4.2.1 Les assertions[ea05a]

≪ Une assertion est un ensemble d'informations qui fournit zéro ou plusieurs déclarations faites par une autorité SAML. ≫L'autorité est donc la partie qui affirme quelque chose et est donc parfois mentionnée comme une *asserting party*, les entités qui les utilisent sont elles appelées *relying parties*.

Une assertion concerne en général un sujet, représenté par l'attribut *Subject* dans une assertion. Cet attribut est toutefois optionnel, il est donc possible de créer des assertions sans préciser un sujet ou en le faisant d'une autre manière. Les relying parties utilisant les assertions sur un sujet afin de contrôler l'accès à une ressource jouent sont appelées *Service Provider*. L'*asserting paty* qui leurs fournit ces assertions est appelée l'*identity provider*.

Les assertions déclarant une information sont toujours liées à un sujet. Ces déclarations peuvent être de trois types différents :
- authentification
- attributs
- décision d'autorisation

La structure d'une assertion est générique et fournit des informations communes à toutes les déclarations. D'autres éléments sont plus spécifiques suivant le type de l'assertion.

Les identifiants tiennent une place importante dans SAML. Ils sont utilisés afin d'identifier les sujet mais également l'émetteur d'une assertion.

Une assertion est constituée d'un ensemble d'éléments et d'attributs. Toute assertion requiert 3 attributs qui sont la version de SAML utilisée, un identifiant et le moment ou a été produite l'assertion. A cela il faut ajouter au moins un élément qui est l'émetteur. On peu ajouter ensuite différents éléments comme une signature, un sujet, des conditions, des conseils et bien sur les différents types de déclarations. Une assertion peut être cryptée. Elle est alors constituée de deux éléments, l'assertion cryptée proprement dite et si nécessaire une ou plusieurs clefs de décryptage.

Le sujet est l'élément central d'une assertion. Il est constitué obligatoirement d'un identifiant et si nécessaire d'un moyen de confirmer l'identité du sujet. Cette confirmation peut être faite à l'aide de différentes méthodes.

Les déclarations constituent un autre élément important d'une assertion. Les trois types de déclarations présentées exigent tous qu'un sujet soit présent dans l'assertion.

La déclaration d'authentification requiert deux éléments, le moment d'authentification et le contexte d'authentification. D'autre éléments peuvent s'y ajouter comme un index de session, le moment d'expiration de cette session ou la localisation du sujet. Le contexte d'authentification permet de fournir des informations additionnelles afin d'évaluer le niveau de confiance qu'une entité peut avoir dans une assertion. Ces informations peuvent être le mécanisme d'i-

dentification initial, les mécanismes utilisés pour minimiser les risques que les informations d'identité soient compromises, pour les stocker et les protéger et le mécanisme d'authentification. Il existe potentiellement beaucoup de contexte différent. La spécification définit donc une liste de classe qui représentent les pratiques et technologies courantes pour l'authentification.

La déclaration d'attributs est constituée d'un ou plusieurs attributs. Les attributs peuvent être cryptés de la même manière que pour les assertions. Un attribut est identifié par un nom. A cela on peut ajouter le format utilisé pour le nom de l'attribut ou un nom plus facilement compréhensible. Enfin on peut ajouter une ou plusieurs valeurs pour cet attribut. L'absence de valeur ayant parfois une signification particulière suivant le profil utilisé. Le type de valeur peut être de n'importe quel type XML bien formé, mais on utilise en général les types simples d'XML Schema.

Pour la déclaration de décision d'autorisation, il est conseillé d'utiliser XACML, la spécification de la déclaration de décision ayant été gelée à partir de la version 2 de SAML. De plus, XACML propose des fonctionnalités de décision d'autorisation plus évoluées comme nous le verrons dans une section qui lui est consacrée un peu plus bas. Une spécification définit clairement comment utiliser XACML et SAML ensemble. La définition de la déclaration de décision d'autorisation sera remplacée par une autre permettant de transmettre une réponse XACML.

4.2.2 Les protocoles

Les messages de protocoles SAML peuvent être échangés via différents protocoles[ea05a]. Les protocoles permettent de réaliser différentes actions tel que obtenir une assertion, demander une authentification ou au contraire la déconnexion à une session.

Requêtes et réponses

Tous les protocoles sont basés sur l'échange de requêtes et de réponses. Une requête contient nécessairement un identifiant, la version de SAML utilisée et le moment ou elle a été produite. D'autres élément peuvent être précisés comme l'émetteur, une signature, la destination du message ou le consentement de l'utilisateur. Pour la réponse, la structure est quasi identique à celle de la requête. Pour une réponse, il est nécessaire de préciser le statut de la requête. On peut également préciser l'identifiant de la requête pour laquelle la réponse a été émise. Le statut comprend un code et si nécessaire un message et des informations additionnelles. Le code lui même comprend une valeur et éventuellement d'autres codes permettant de préciser la nature de l'erreur. La valeur du code indique le succès de la requête ou dans le cas contraire si le problème provient du demandeur ou du répondeur. Une version incorrecte du message peut également être une valeur possible. Un ensemble de valeur de second niveau sont également utilisées pour exprimer de manière plus précise la nature de l'erreur survenue.

Protocole de requête et de demande d'assertions

Le protocole de requête et de demande d'assertions défini les messages et processus à utiliser pour demander ou requérir une assertion. Une requête se fait par rapport à la référence d'une assertion, les demandes se font par rapport au sujet. Pour un sujet connu, on peut demander les assertions d'authentification ,d'attributs ou de décision d'autorisations qui ont déjà été émise pour ce sujet.

Protocole de requête d'authentification

Lorsqu'un utilisateur désire établir un contexte de sécurité chez une relying party, il utilise le protocole de requête d'authentification afin d'obtenir un assertion d'authentification. Ce protocole fait intervenir différents acteurs :

- le demandeur est l'entité qui demande l'authentification et à laquelle la réponse est renvoyée
- le présentateur est l'entité qui présente la requête à l'identity provider
- le sujet demandé est l'entité pour laquelle l'assertion est demandée
- l'asserting prty est l'entité capable de confirmer l'identité du sujet
- la relying party est l'entité qui consomme l'assertion
- l'identity provider est l'entité à qui le présentateur donne la requête et duquel il reçoit la réponse

Protocole de Single Logout

Le protocole de Single Logout permet de terminer un ensemble de session en même temps auprès d'une autorité de session. La demande peut être faite aussi bien auprès de l'autorité qu'auprès d'une entité participant à la session. Le protocole peut également être utilisé directement par l'autorité afin de terminer la session d'un utilisateur à cause d'un timeout par exemple.

Signature et chiffrement

Certaines assertions ou certains messages peuvent être signés. La signature peut apporter des avantages en terme d'intégrité, d'authentification et de non-répudiation. Les messages ne doivent pas nécessairement être signé directement si une signature peut être hérité par l'établissement d'un canal de communication sécurisé entre deux entités. Lorsque c'est impossible, les assertions et les messages peuvent être signés via XML Signature en utilisant l'élément `<ds:Signature>` .

Le chiffrement est un moyen de garantir la confidentialité. La communication en elle-même peut être cryptée via l'utilisation de SSL/TLS. Sinon les assertions, attributs et identifiants peuvent être cryptés à l'aide de XML Encryption. Ces éléments sont alors remplacés par l'élément crypté dans le message.

4.2.3 Les bindings[ea05b]

Les bindings définissent la manière dont l'échange des messages SAML peuvent être effectué via des messages et protocoles de communication standard

tels que SOAP et HTTP.

SOAP Binding

Les messages SOAP sont constitué d'une enveloppe, d'une en-tête et d'un corps. Les éléments SAML seront inclus dans le corps du message SOAP. L'utilisation de SOAP se fait sur le modèle de requête-réponse. L'émetteur envoi une seule requête SAML dans le corps d'un message SOAP er reçoit une seule réponse dans le corps d'un message SOAP.

L'utilisation de SAML par l'utilisation de messages SOAP exige l'utilisation de HTTP pour transporter les messages SOAP. L'utilisation de SOAPAction dans l'en-tête HTTP est requise.

Reverse SOAP binding

Dans ce binding, l'échange des messages se fait également via SOAP sur HTTP. L'échange n'est cependant plus basé sur le modèle simple de requête-réponse comme pour SOAP. Ce binding est une réponse au profile ECP que nous verrons plus tard. Un client joue le rôle d'intermédiaire. L'échange des messages se fait via deux requêtes HTTP. Une première requête HTTP est envoyée par le client qui reçoit une réponse HTTP et c'est cette réponse qui contient la requête SAML dans un message SOAP. La réponse à cette requête sera envoyée via un message SOAP mais dans une requête HTTP. Le client recevra bien évidement une réponse HTTP à sa requête initiale. C'est dans ce sens qu'il est appelé reverse SOAP (POAS) car une requête SAML se via un réponse HTTP et une réponse SAML via un requête HTTP.

HTTP binding

Les différents bindings utilisant HTTP sont HTTP Redirect binding, HTTP Post binding et HTTP Artifact binding. Ces bindings sont utilisés quand le demandeur ne partage pas un canal de communication direct avec le répondeur. L'utilisation d'un agent HTTP comme intermédiaire est alors nécessaire. Ces bindings permettent de transmettre les messages SAML du demandeur au répondeur en passant par un agent HTTP qui joue le rôle d'intermédiaire. Les différents bindings HTTP peuvent être composés afin d'envoyer un même message via deux bindings.

Redirection HTTP

La transmission des messages SAML se font via les paramètres inclus dans l'URL. Les messages SAML étant de l'XML, ils peuvent être encodés de différentes manières dans une URL.

POST binding

Les messages SAML donc XML sont encodés dans un formulaire HTTP et envoyés via la méthode HTTP POST. Le message SAML est encodé en appliquant les règles d'encodage base-64 à la représentation XML du message. Le message ainsi encodé est placé dans un formulaire caché.

Artifact binding

Lorsque l'utilisation d'un agent intermédiaire est nécessaire mais que la transmission du message entier pose problème, il est possible de transmettre les message via une référence appelée artifact. Seul l'artifact est transmis via l'agent, le récepteur utilise alors l'artifact pour obtenir le message entier auprès de l'émetteur via un autre binding tel que SOAP.

4.2.4 Les profiles[ea05c]

Dans cette section, nous présenterons l'aperçu des différents profiles afin de pouvoir identifier les différents acteurs et leurs interactions.

Web Browser SSO Profile

Ce profil supporte 2 scénarios, soit l'utilisateur accède à une ressource chez un Service Provider, soit il accède à l'Identity Provider, la ressource et le Service Provider étant sous-entendu ou implicite.

L'idée principale est que l'Identity Provider émet une assertion d'authentification que le Service Provider utilise afin d'établir un contexte de sécurité pour l'utilisateur.

La spécification suppose que l'utilisateur utilise un navigateur standard et qu'il possède un moyen de s'authentifier auprès de l'Identity Provider.

Les différentes étapes représentées sur la figure 4.2 sont expliquées ci-dessous.

1. Le User-AGent tente d'accéder à une ressource du Service Provider
2. Le Service Provider détermine l'Identity Provider à utiliser
3. Le Service Provider émet une AuthnRequest à destination de l'Identity Provider via l'User-Agent
4. L'Identity Provider authentifie l'utilisateur
5. L'Identity Provider émet une réponse à destination du Service Provider via l'User-Agent
6. Le Service Provider autorise ou refuse l'accès à l'utilisateur

La manière dont l'utilisateur est authentifié par l'Identity Provider est hors du cadre de SAML.

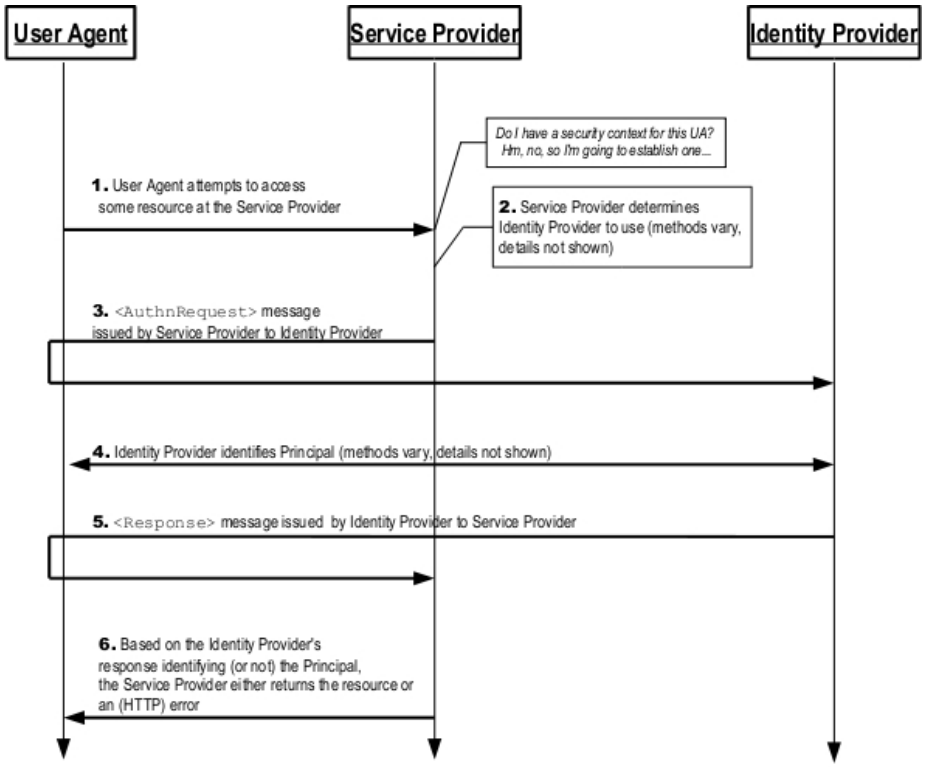

FIGURE 4.2 – Profil Web Browser SSO

Enhanced Client or Proxy (ECP) Profile

Un ECP est une entité qui sait comment contacter l'Identity Provider approprié et supporte le Reverse SOAP(POAS).

Ce profile supporte le même scénario que le Web Browser SSO mais pour un client ou un proxy. Il est présenté à la figure 4.3 et décrit ci-dessous.

1. L'ECP émet une requête HTTP au Service Provider
2. Le Service Provider émet une *AuthnRequest* à l'ECP
3. L'ECP détermine l'Identity Provider à contacter
4. L'ECP transmet l'*AuthnRequest* à l'Identity Provider
5. L'Identity Provider authentifie l'utilisateur
6. L'Identity Provider émet une *Response* à l'ECP pour le Service Provider
7. L'ECP transmet la *Response* au Service Provider
8. Le Service Provider autorise ou refuse l'accès à l'utilisateur

La manière dont l'Identity Provider authentifie l'utilisateur est hors du cadre de SAML.

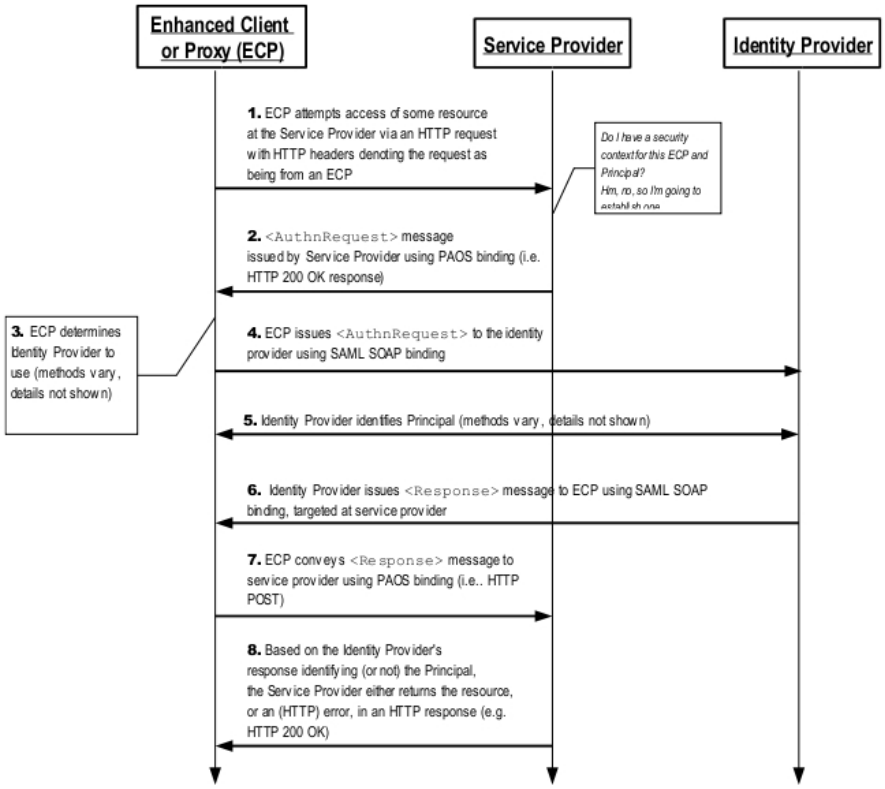

FIGURE 4.3 – Profil Enhanced Client or Proxy (ECP)

Single Logout Profile

L'Identity Provider peut agir comme l'autorité de session et les différents Service Providers comme participants à la session. Si l'utilisateur désire terminer sa session avec tous les participants, il peut utiliser ce profil.

Les différentes étapes représentées sur la figure 4.4 sont expliquées ci-dessous.

1. Une *LogoutRequest* émit par un Session Participant à l'Identity Provider
2. L'Identity Provider détermine les entités participante à la session
3. L'Identity Provider émet une *LogoutRequest* au Session Participant
4. Le Session Participant émet une *LogoutResponse* à l'Identity Provider
5. L'Identity Provider émet une *LogoutResponse* au Session Participant

La première et la cinquième étape peuvent être ignorées si la fin de session est à l'initiative de l'Identity Provider. Les étapes 3 et 4 sont répétées pour tous les participants à la session.

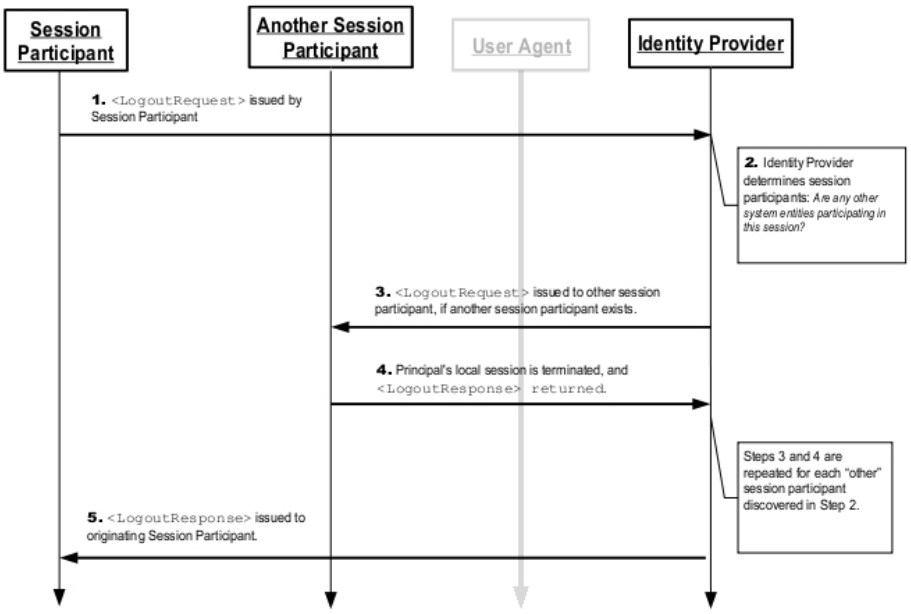

FIGURE 4.4 – Profil Single Logout

Assertion Query/Request Profil

Le profil de demande/requête d'assertion correspond au protocole de requête et de demande d'assertions décrit dans la section 4.2.2. Le profil est assez simple :

1. le demandeur émet une requête auprès de l'autorité
2. l'autorité émet une réponse au demandeur

Le profile impose l'utilisation de SOAP comme binding et exige que l'émetteur de la requête et de la réponse soit précisé.

4.3 OpenID[Ope07]

OpenID est un système d'authentification décentralisé qui permet l'authentification unique, ainsi que le partage d'attributs. Cette authentification se fait sans que l'utilisateur doivent fournir des informations sensibles telles que un mot de passe ou une adresse e-mail à l'entité auprès de laquelle il désire s'authentifier.

Les entité nécessaire au protocole sont l'*OpenID Provider*, le *Relying Party* et l'*User-Agent*. L'*OpenID Provider* n'est rien d'autre que le serveur d'authentification OpenID, le *Relying Party* est l'application web qui demande l'authentification de l'utilisateur et l'*User-Agent* est le navigateur de l'utilisateur. Tous les messages sont échangés via des requêtes et réponses HTTP.

Le protocole est assez simple. L'utilisateur commence par présenter un identifiant à un *Relying Party* via son *User-Agent*. A l'aide de cet identifiant, le *Relying Party* découvre l'*OpenID Provider* qui y est associé. Le *Relying Party* et

l'*OpenID Provider* peuvent établir une association en partageant un secret. Cela permettra au RP d'éviter de devoir vérifier les assertions directement auprès de l'*OpenID Provider*. En effet, grâce au secret partagé, l'*OpenID Provider* peut signer les assertions et le *Relying Party* a ainsi une preuve que les assertions ont bien été produites par l'*OpenID Provider*. Le *Relying Party* redirige ensuite l'utilisateur vers l'*OpenID Provider* avec une requête d'authentification. L'*OpenID Provider* authentifie alors l'utilisateur et le redirige à son tour vers le *Relying Party* avec une assertion d'authentification. Le *Relying Party* reçoit cette assertion et la vérifie soit grâce à une signature, soit directement auprès de l'*OpenID Provider*.

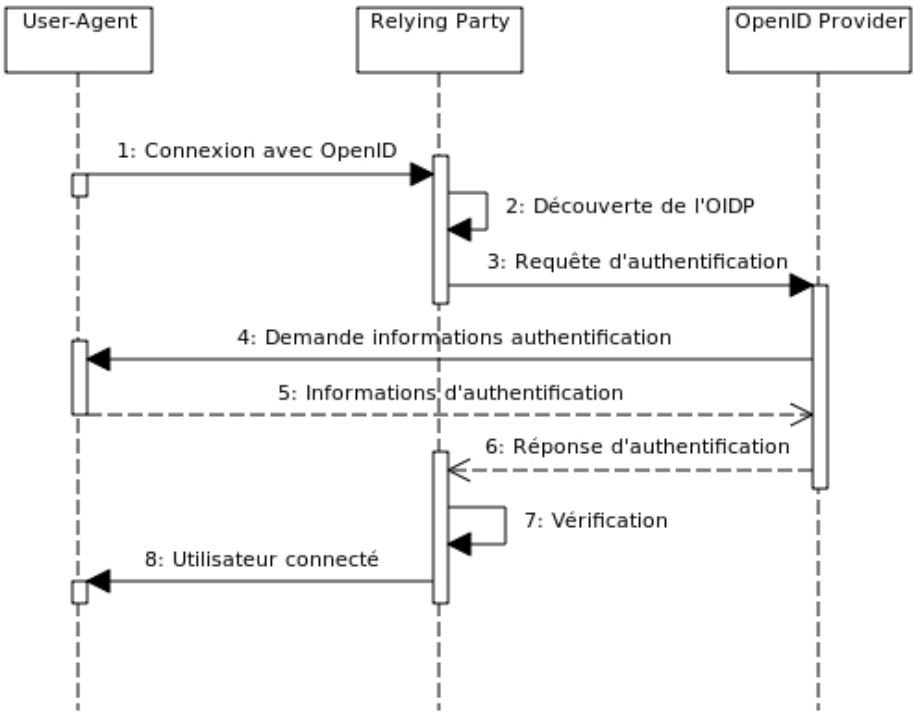

FIGURE 4.5 – Protocole OpenID

Les messages OpenID consiste en une association de clefs et de valeurs. Les messages peuvent être encodés de différentes manières :

Encodage clef-valeur Le message est encodé sous la forme d'une séquence de lignes. Chaque ligne est constituée dans l'ordre d'une clef, deux points(:), une valeur et le symbole d'une nouvelle ligne(`\n`).

Encodage HTTP Le message est encodé sous la forme d'un formulaire HTML. Chaque couple clef/valeur est séparé par un `&` et la clef est séparée de la valeur par le signe `=` . Chaque nom de clef sera précédé par "`openid.` ". Les messages envoyés via une requête HTTP nécessite 2 champs, openid.ns avec la valeur "http ://specs.openid.net/auth/2.0" et openid.mode avec comme valeur le type de message.

Il existe deux types de communication. La communication directe pour établir une association et vérifier les assertions d'authentification. Les demandes doivent être encodées dans le corps d'un message POST HTTP comme décrit dans l'encodage HTTP. Les réponses sont encodées via l'encodage clef-valeur et doivent contenir le champs ns avec comme valeur http ://specs.openid.net/auth/2.0". Pour la communication indirecte, les messages sont échangés en passant par un agent. Elle est utilisée pour la demande d'authentification et pour la réponse. Les messages sont encodés via l'encodage HTTP et peuvent être envoyés soit via une redirection HTTP, soit par la redirection d'un formulaire. Dans les deux cas, l'émetteur doit connaitre l'adresse du récepteur auquel il destine les messages.

L'association entre un *OpenID Provider* et un *Relying Party* est généralement réalisée par l'utilisation d'un Message Authentication Code (MAC) comme clef pour signer les messages échangés.

4.4 OAuth

OAuth est « un protocole qui permet d'une méthode simple et standard d'autoriser l'accès à une application bureau ou web sécurisée. »[HL10] OAuth permet soit d'accéder aux ressources d'un serveur au nom du propriétaire de la ressource, soit à un utilisateur de partager ses ressources avec quelqu'un d'autre sans lui fournir ses informations de sécurité(mot de passe).

Pour obtenir l'accès à une ressource, un client doit en obtenir la permission du propriétaire. Cette permission est exprimée sous forme d'un jeton. Ce jeton peut être donné en y incluant certaines restrictions sur son champ d'application ou sa durée de vie et il peut être révoqué à tout moment.

Le processus d'autorisation basé sur la redirection s'effectue en 3 étapes :
- Le client obtient un certificat temporaire du serveur qui identifie la demande d'accès
- Le propriétaire de la ressource autorise le serveur à accepter la demande d'accès
- Le client utilise son certificat afin de récupérer les jetons qui lui permettront d'accéder aux ressources protégées.

4.5 XACML

eXtensible Access Control Markup Language(XACML) est un standard qui défini à la fois un langage de politique de sécurité et un langage pour échanger les décisions de contrôle d'accès.[Mic03] Le langage permet de décrire les exigences qui sont en général nécessaires pour le contrôle d'accès mais possède de nombreuses extensions afin de pouvoir définir une politique complète et en adéquations avec les exigences désirées. Le langage d'échange de décisions permet de demander un accès et d'en obtenir une réponse.

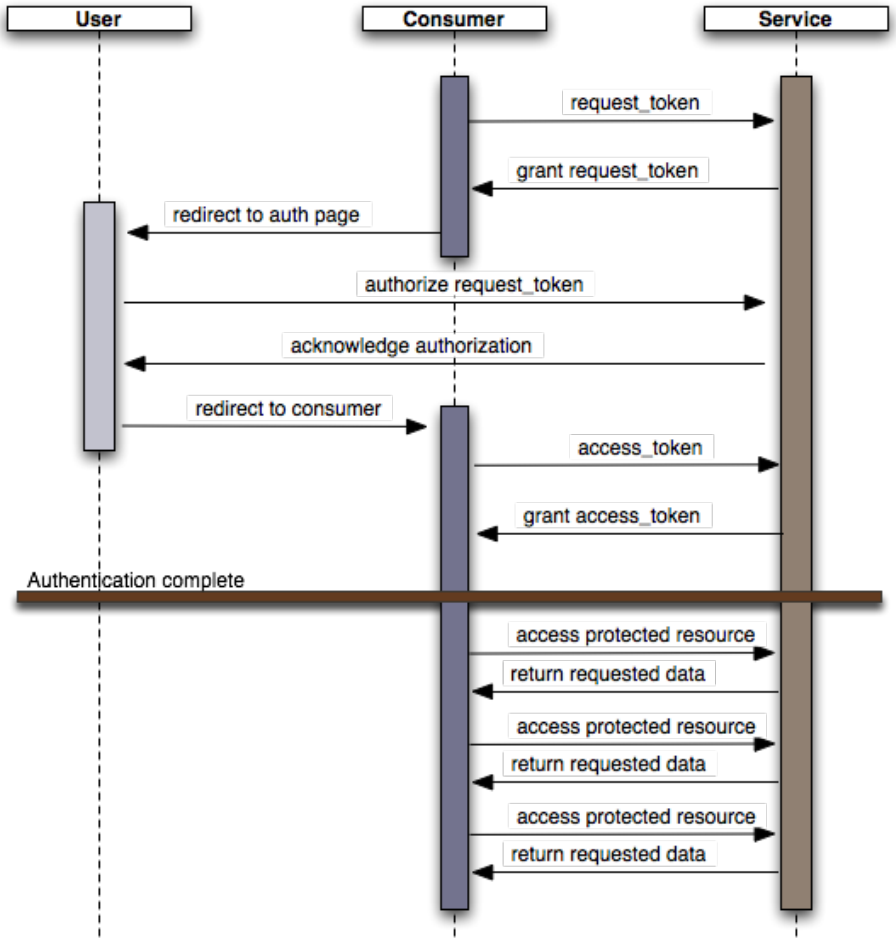

FIGURE 4.6 – Protocole OAuth[OAu]

D'autres langages permettent d'exprimer tout cela mais XACML permet d'avoir un standard examiné et approuvé par une communauté d'experts, est générique et ne dépend donc pas de l'environnement ni de l'application. Une même politique peut être utilisées pour différents types d'applications. La politique peut être divisée et chaque partie gérée par différentes personnes. XACML se charge ensuite de les combiner. Enfin, différents profiles XACML ont été défini afin de pouvoir incorporer XACML dans d'autres standards tel que SAML ou LDAP.

4.5.1 Echange de décisions

Les acteurs majeurs de XACML représentés sur la figure 4.7 sont décrits ci-dessous :

PAP Le Policy Administration Point est l'entité qui crée les politiques de sécurité des droits d'accès logiques.

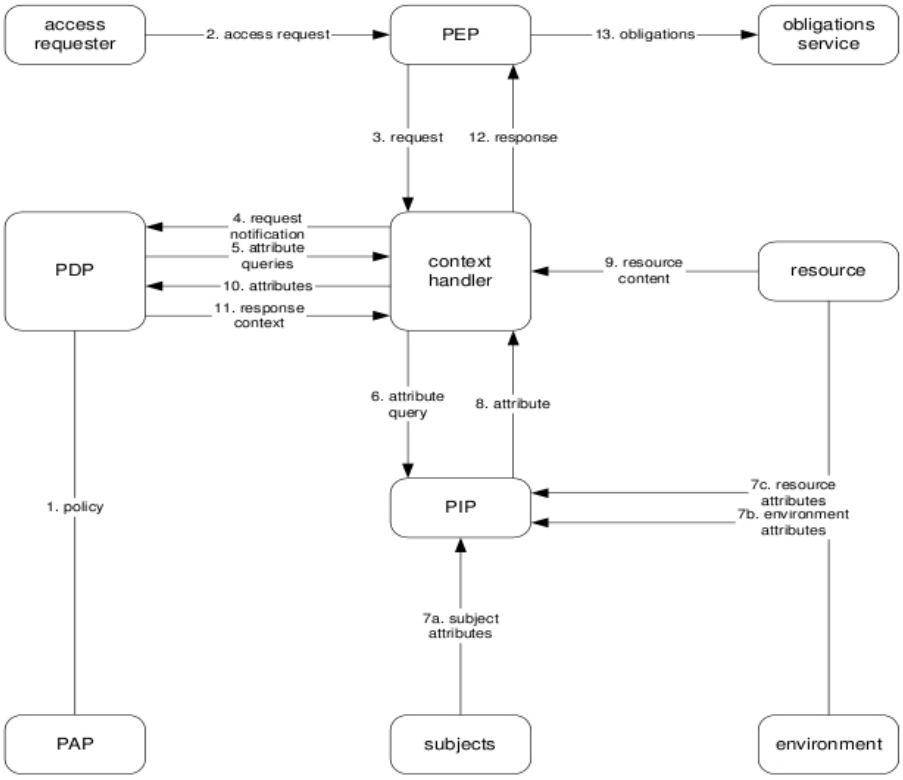

FIGURE 4.7 – Acteurs XACML

PDP Le Policy Decision Point est l'entité qui évalue les politiques applicable et rend une décision d'autorisation.

PEP Le Policy Enforcement Point est l'entité qui accompli le contrôle d'accès en demandant des autorisations et en appliquant les décisions prises.

PIP Le Policy Information Point est la source des attributs.

gestionnaire de contexte Le gestionnaire de contexte est l'entité chargée de convertir les requêtes et autorisations du langage natif à XACML.

Le flux de données également représenté sur la figure 4.7 entre ces différents acteurs sont ci-dessous :

1. PAPs écrivent les politiques et les rendent disponibles auprès des PDPs
2. Le demandeur d'accès émet une requête auprès du PEP
3. Le PEP envoi la requête au gestionnaire de contexte en y incluant éventuellement des attributs sur l'utilisateur, la ressource, l'action ou l'environnement.
4. Le gestionnaire de contexte construit un contexte de requête XACML
5. Le PDP demande au gestionnaire de contexte des attributs additionnels
6. Le gestionnaire de contexte demande les attributs additionnels au PIP
7. Le PIP obtient les attributs additionnels
8. Le PIP transmet les attributs demandés au gestionnaire de contexte

9. Le gestionnaire de contexte peut éventuellement inclure la ressource dans le contexte
10. Le gestionnaire de contexte transmets les attributs demandés au PDP. Le PDP évalue la politique
11. Le PDP retourne le contexte de réponse au gestionnaire de contexte
12. Le gestionnaire de contexte transmet la réponse au PEP
13. Le PEP rempli les obligations
14. Le PEP autorise ou refuse l'accès à la ressource

Dans ce modèle, nous pouvons voir comment les acteurs majeurs échangent les décisions de contrôle d'accès via XACML. Mais XACML peut également être utilisé pour décrire les politiques de sécurité des droits d'accès.

4.5.2 Expression d'une politique de sécurité

Une décision de contrôle d'accès est prise sur une politique complète. Cette politique peut être basée sur un ensemble de règles et de politiques combinée en une politique unique applicable à la requête.

Afin de construire une politique, XACML définit les trois éléments principaux suivants : Rule, Policy et PolicySet.[**?**]

L'élément Rule est le plus basique des éléments. Il contient une expression booléenne. Le PDP peut l'évalué seul mais ne peut y accéder directement. En effet, l'élément seul ne peut pas être seule à la base d'une décision d'autorisation mais est l'unité basique pour construire une politique.

L'élément Policy contient un ensemble de Rule et une procédure pour combiner le résultat de leur évaluation. Il est à la base d'une décision d'autorisation.

L'élément PolicySet contient un ensemble de Policy ou d'autres PolicySet et la procédure pour combiner les résultats de leurs évaluations. C'est le moyen standard pour combiner différentes politiques en une politique unique.

XACML définit un ensemble d'algorithme pour combiner les résultats des évaluations des Rule ou Policy. Les algorithmes standards sont :
- le refus l'emporte
- la permission l'emporte
- la première applicable
- une seule applicable

Il est cependant possible de définir son propre algorithme.

Il est assez fréquent qu'une décision d'autorisation se base sur les caractéristiques d'un sujet et pas seulement son identité. Le contrôle d'accès basé sur les rôle (RBAC) en est un bon exemple. Il arrive également que les attributs d'une ressource ou son contenu influence une décision d'autorisation. XACML permet de supporter ces différentes approches.

XACML permet d'implémenter le modèle RBAC.[Ris10b] Le modèle d'accès basé sur les rôles (RBAC) repose sur les éléments suivants : les utilisateurs, les rôles, les objets, les opérations et les permissions. Chacun de ces éléments peut être implémenté avec XACML. Un utilisateur est implémenté par un Subject, un rôle est exprimé par un ou plusieurs Subject Attributes, un objet par une Ressource, une opération par une Action et les permissions par des Role PolicySet et Permission PolicySet

4.5.3 Profile SAML de XACML

Le *SAML 2.0 profile of XACML v2.0*[RL10] définit un profile qui permet de protéger, transporter et demander les informations nécessaires à l'implémentation de XACML.

Les 4 entités pour lesquelles SAML peut être utilisé comme support de communication sont le Policy Enforcement Point(PEP), l'Attribute Authority(PIP), le Policy Decision Point(PDP) et le Policy Admin Point(PAP). A cela on peut ajouter l'Attribute Repository et le Policy Repsository.

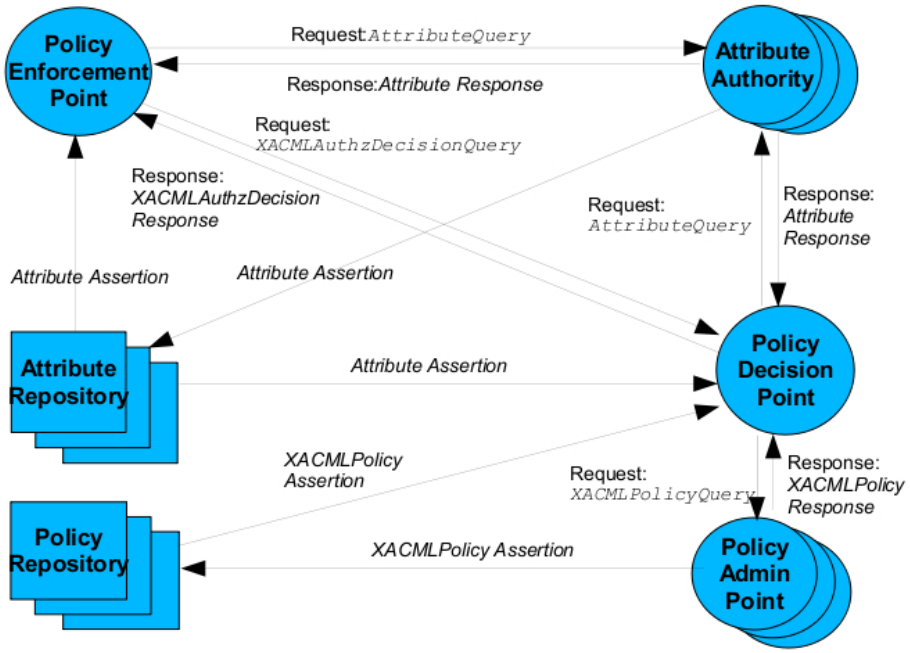

FIGURE 4.8 – Composants et messages dans une intégration de SAML avec XACML

Comme on peut le voir sur la figure 4.8, les attributs et demandes d'attributs peuvent facilement être transporté via des messages SAML. De même, les décisions d'autorisations peuvent également être demandées et transmises via des messages SAML.

L'échange de politiques XACML peut également se faire via SAML. Le profile définit une extension de type et d'élément et décrit comment ils sont utilisés avec les autres éléments standards de SAML. Le `XACMLPolicyStatementType` est une extension du `Statement` SAML. Les politiques seront donc échangées via des `Statement` de type `XACMLPolicyStatementType`. Ces `Statement` seront transportés dans des assertions SAML.

Afin de demander un politique XACML à une entité, on utilisera une `XACMLPolicyQuery`. Afin de répondre à cette demande, on utilisera un réponse SAML classique qui contiendra une assertion contenant elle même un `Statement` de type `XACMLPolicyStatementType`. On peut également ajouter qu'un XACMLAuthzDecision Statement peut être utilisé comme un jeton d'autorisation. Ce jeton permet à un client de transmettre un décision d'autorisation provenant d'un tiers de confiance à un service.

4.6 Shibboleth[Intb]

Shibboleth est un projet de l'internet2 dont le but est de proposer un package open source pour le web SSO à l'intérieur ou hors des frontières de l'organisation. Il permet aux différents sites de prendre des décisions d'autorisation grâce à l'échange d'attributs tout en proposant des fonctions permettant de préserver la vie privée. Afin de fournir un cadre à la propagation d'attributs, Shibboleth implémente le standard SAML.

Le système Shibboleth est constitué de deux parties, l'*Identity Provider* qui connait les utilisateurs désirant accéder à un service protégé et le *Service Provider* qui gère le service protégé. On peut ajouter à cela le navigateur et en option le WAYF(Where are ou from ?) qui à pour but d'orienter l'utilisateur vers son IdP.

Etant principalement basé sur SAML, le fonctionnement est similaire au profile de WebSSO expliqué en détail dans la section 4.2.4. L'utilisateur visite un Service Provider. Celui requiert une authentification et le redirige vers l'Identity Provider. L'Identity Provider répond avec un assertion d'authentification et des attributs.

Il est cependant interessant d'examiner l'architecture logique des deux acteurs principaux, le Service Provider et l'Identity Provider décrit dans [OS05]. Le fournisseur de services est composé de trois briques : le consommateur d'assertion, le demandeur d'attributs et le controleur d'accès.

Le consommateur d'assertion est chargé de vérifier que l'utilisateur est bien authentifié par la consommation d'une assertion d'authentification, comme son nom l'indique. Si l'utilisateur n'est pas authentifié, il le redirige vers l'Identity Provider. Une fois la vérification effectuée, il transmet au demandeur d'attributs l'identité de l'utilisateur.

Le demandeur d'attributs est chargé de récupérer les attributs de ses utilisateurs auprès de l'Identity Provider. Ces attributs seront fournis à un contrôleur

d'accès.

Le contrôleur d'accès est chargé d'autoriser ou de refuser l'accès aux ressources demandées.

L'architecture du fournisseur d'identité comprend également trois briques : le service d'authentification, l'autorité d'authentification et l'autorité d'attributs.

Le service d'authentification est chargé d'authentifier l'utilisateur en lui demandant de fournir une information qu'il vérifie ensuite. Typiquement, l'utilisateur fournira un couple identifiant et mot de passe qui sera vérifié auprès d'une base d'authentification du SI. Le service d'authentification ne fait pas partie de l'IdP selon les spécifications de Shibboleth mais est un service nécessaire à l'IdP. Le service d'authentification fournit l'identifiant de l'utilisateur à l'autorité.

L'autorité associe cet identifiant avec l'identifiant qu'il communiquera aux fournisseurs de services.

L'autorité d'attributs est chargée de fournir les attributs d'un utilisateur suite à une demande d'un fournisseur de services. Les attributs sont récupérés auprès des différentes sources du système d'information. L'association maintenue par l'autorité permet de récupérer des attributs à l'aide de l'identifiant réel de l'utilisateur correspondant à l'identifiant fournit par le fournisseur de services.

Le service d'authentification est nécessaire, comme expliqué plus haut mais ne doit pas obligatoirement être réalisé par l'IdP. Ce service peut être réalisé par un système de SSO tel que CAS dont nous décrivons le fonctionnement dans la section 4.7.

4.7 CAS[Maz]

Central Authentication Service (CAS) est un système d'authentification unique (SSO) pour les applications web.

L'architecture est composée de trois acteurs principaux : le serveur CAS, le navigateur et le client CAS. Le serveur CAS authentifie les utilisateurs et transmet et certifie leurs identités aux clients CAS. Le client CAS détient une ressource qu'il ne transmet qu'après qu'il se soit assuré que l'utilisateur a bien été authentifié auprès du serveur CAS.

Le CAS est accessible via 3 URL : l'URL de login, l'URL de validation et l'URL optionnelle de logout. Le fonctionnement de CAS est le suivant : Un utilisateur tente d'accéder à une ressource via son navigateur auprès d'un client CAS. Le client désire connaitre l'identité de l'utilisateur afin de lui délivrer ou non la ressource. Pour cela, il redirige l'utilisateur vers la login URL du serveur CAS. L'utilisateur peut alors s'authentifier auprès du serveur CAS. Le CAS renvoi un "ticket-granting cookie" au navigateur, ce cookie est optionnel mais si

il est accepté permettra de fournir un service de SSO à l'utilisateur. Lors de la redirection, le service a fourni son identifiant. Le CAS va utiliser cet identifiant pour générer un ticket de service. Ce ticket n'est utilisable qu'une seule fois et uniquement par le service qui en a fait la demande. Le serveur CAS va alors rediriger l'utilisateur vers le service d'origine avec le ticket généré. Le service va ensuite vérifier ce ticket directement auprès du serveur CAS via l'URL de validation. Si la vérification réussi, le service peut considérer l'utilisateur comme authentifié et lui délivrer la ressource.

Si un utilisateur désire accéder à une ressource d'un autre client CAS, ce client le redirigera vers le serveur CAS. Si il a accepté le "ticket-granting-cookie", il obtient directement un ticket de service en présentant son ticket-granting-cookie sans devoir se re-authentifier auprès du serveur CAS.

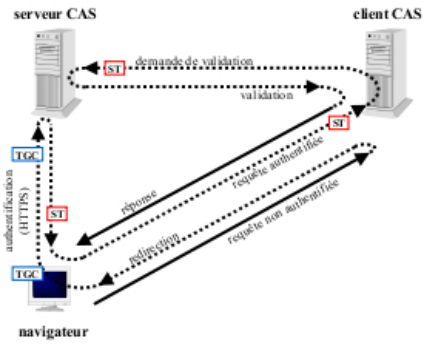

FIGURE 4.9 – Service d'authentification CAS[VM]

L'intérêt de CAS est qu'il est capable de fonctionner dans un environnement multi-tiers[VM]. Un mandataire CAS doit parfois accéder à un client CAS au nom de cet utilisateur. Pour cela, il peut demander après le vérification d'un ticket de service que lui fournit un utilisateur, un "Proxy Granting Ticket" (PGT). Ce PGT est en quelques sorte l'équivalent d'un "ticket-granting cookie" du navigateur. Il permet au mandataire d'obtenir des "Proxy Ticket" qui sont l'équivalent d'un "Service Ticket". Le mandataire peut donc accéder à des ressources au nom d'un utilisateur en fournissant des "Proxy Ticket" obtenus auprès du serveur CAS sans devoir faire appel à l'utilisateur à chaque fois.

4.8 LemonLDAP : :NG

"LemonLDAP : :NG est un WebSSO modulaire basé sur les modules Apache : :Session."[Lemb] Il propose une protection complète d'un espace web avec Authentification, Autorisation et Audit. L'architecture est composée de 3 composants principaux : Manager, Portal et Handler.

Le composant principal, le Portal, nécessite 4 modules pour fonctionner.[Lema] Le module Authentication permet de vérifier les credentials des utilisateurs.

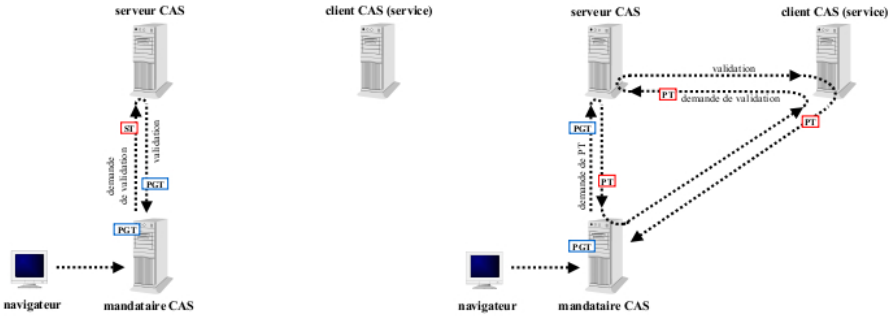

FIGURE 4.10 – Authentification CAS pour un tiers[VM]

Pour cela, il peut faire appel à différents systèmes en arrière plan tel que (LDAP, Apache (Kerberos), CAS, OpenID, SAML/Shibboleth). Le second module, le User database, permet de collecter différentes informations sur les utilisateurs. Pour cela, il peut également faire appel à d'autres systèmes comme LDAP, les bases de données classiques(MySQL, Oracle, PostGreSQL), OpenID ou SAML/Shibboleth. Le module Password database permet de gérer les mots de passe. Il peut également faire appel à LDAP ou les bases de données classiques. Enfin, le module Identity Provider permet de propager l'identité d'un utilisateur. Pour ce faire, il peut utiliser les services d'Identity Provider de SAML, OpenID ou CAS.

4.9 Synthèse des standards, protocoles et architectures

Afin de proposer un solution complète de contrôle d'accès, il est nécessaire de pouvoir offrir au moins un service d'authentification et d'autorisation. Kerberos, OpenID et SAML permettent l'authentification. SAML, XACML et OAuth peuvent être utilisés pour gérer les autorisations. XACML permet également l'expression d'une politique de sécurité. Enfin, Shibboleth, CAS et LemonLDAP sont des système de WebSSO.

Une fonctionnalité d'authentification unique est proposée par la plupart des solutions proposées. Cependant, SAML est le seul à proposer l'échange de messages d'authentification, d'attributs et d'autorisation. Pour l'expression de demandes et décisions d'autorisations, il est conseillé d'utiliser XACML dans SAML. De plus, SAML a été conçu dans un souci d'interopérabilité et est utilisé dans différentes solutions de WebSSO. SAML définit également plusieurs profils qui reprennent les scénarios les plus courants. Toutefois, il est possible d'étendre SAML afin qu'il réponde plus précisément à des besoins spécifiques. La combinaison de SAML et XACMl semble être donc la solution la plus complète pour offrir un service de contrôle d'accès. Elle permet la transmission de l'authentification, des attributs et des autorisations.

Les différents systèmes de WebSSO ne permettent pas d'intégrer facilement

les applications non-web, ils ne peuvent donc pas être utilisés dans notre situation. Néanmoins, l'architecture logique peut être une source d'inspiration pour concevoir notre service de contrôle d'accès.

Chapitre 5

Solution

5.1 Architecture globale

Sur base des exigences portant sur le contrôle d'accès, nous pouvons établir une architecture globale. Le service de contrôle d'accès sera donc unique et externe aux applications. De plus, toutes les applications devront pouvoir l'utiliser. Les décisions de contrôle d'accès devront être prises sur base de référentiels centraux (utilisateurs et politiques). Enfin, l'audit et l'administration devront être proposés en plus des services nécessaires au contrôle d'accès (authentification et autorisation). La solution proposée sera donc constituée des différentes applications à protéger (Web et non-web), de l'application d'administration, du service de contrôle d'accès, des différents référentiels(utilisateurs et politiques) et des journaux d'audit.

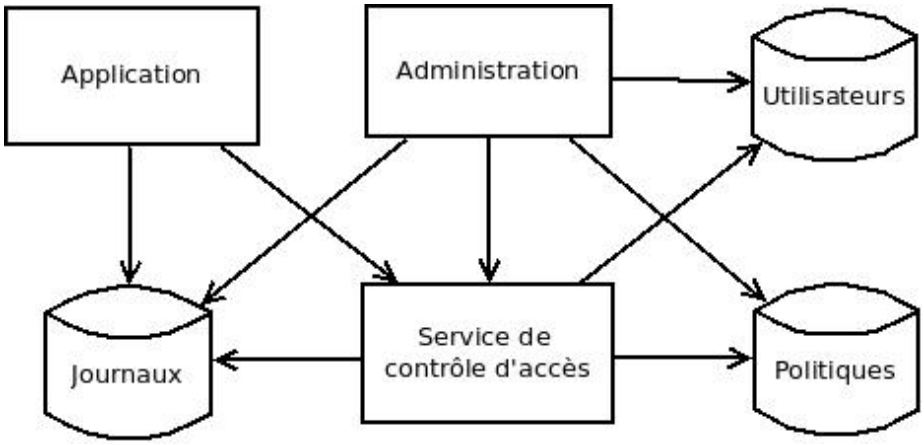

FIGURE 5.1 – Architecture générale

5.1.1 Applications

Les applications utilisent le service de contrôle d'accès afin de protéger leurs ressources. Elles utilisent les fonctionnalités d'authentification et d'autorisation

pour le contrôle d'accès. La fonction d'audit est utilisée afin qu'elles puissent remplir leurs obligations en traçant les accès et les actions effectuées par les utilisateurs.

5.1.2 Application d'administration

L'application d'administration est une application comme toutes les autres, elle fera donc appel au service de contrôle d'accès afin de protéger ses services et ses ressources. Elle sera la seule interface avec les référentiels contenant les utilisateurs et les politiques de sécurité. Les administrateurs seront chargés de gérer les utilisateurs et le système à partir de cette application. Le responsable de la sécurité pourra définir des politiques de sécurité des droits d'accès logiques via cette application et vérifier qu'elles sont bien appliquées en consultant les différents journaux.

5.1.3 Service de contrôle d'accès

Le service de contrôle d'accès est composé de trois grands services : l'authentification , l'autorisation et l'audit. Le service d'authentification sera chargé de fournir aux applications l'identité de leurs utilisateurs. Le service d'autorisation sera chargé de prendre une décision pour l'application en fonction du contexte de la demande de l'application, du profil de l'utilisateur et de la politique à appliquer. Le service d'audit devra quand à lui permettre de tracer tous les événements importants qui surviennent lors du contrôle d'accès.

5.1.4 Référentiels

Le référentiel d'utilisateurs contient toutes les informations nécessaires pour prendre une décision de contrôle d'accès (identité, rôles, attributs, ...). Le référentiel des politiques contient toutes les politiques de sécurités définies par le responsable de la sécurité.

5.1.5 Journaux

Les journaux contiennent toutes les traces des événement survenus lors d'un scénario de contrôle d'accès.

5.2 Scénarios

5.2.1 Contrôle d'accès à une application sécurisée

Le scénario général est le suivant. Un utilisateur désire accéder à une application sécurisée et effectuer des traitements sur des données. L'utilisateur est connu de l'hôpital, il possède un badge et/ou un couple identifiant/mot de passe. L'application est propre à l'hôpital. Le scénario se déroule donc dans le

domaine unique de Saint-Luc. L'hôpital possède une politique de sécurité des droits d'accès logiques qui restreint l'accès aux applications et aux informations sensibles. Lorsque l'utilisateur effectuera une requête auprès de l'application, celle-ci devra vérifier si il peut accéder ou non à ses services et ses ressources. Pour cela, elle va commencer par authentifier l'utilisateur. Ensuite, sur base de l'identité de cet utilisateur, elle demandera une autorisation pour lui. Elle pourra également demander des informations supplémentaires sur l'utilisateur. Le service de contrôle d'accès permettra de certifier de l'identité de l'utilisateur et de fournir des autorisations et des informations supplémentaires aux applications. La décision d'autorisation sera basée sur la politique de sécurité applicable, le contexte dans lequel l'accès est demandé et le profil de l'utilisateur. Tous les événements qui surviennent lors de ce scénario devront être tracés.

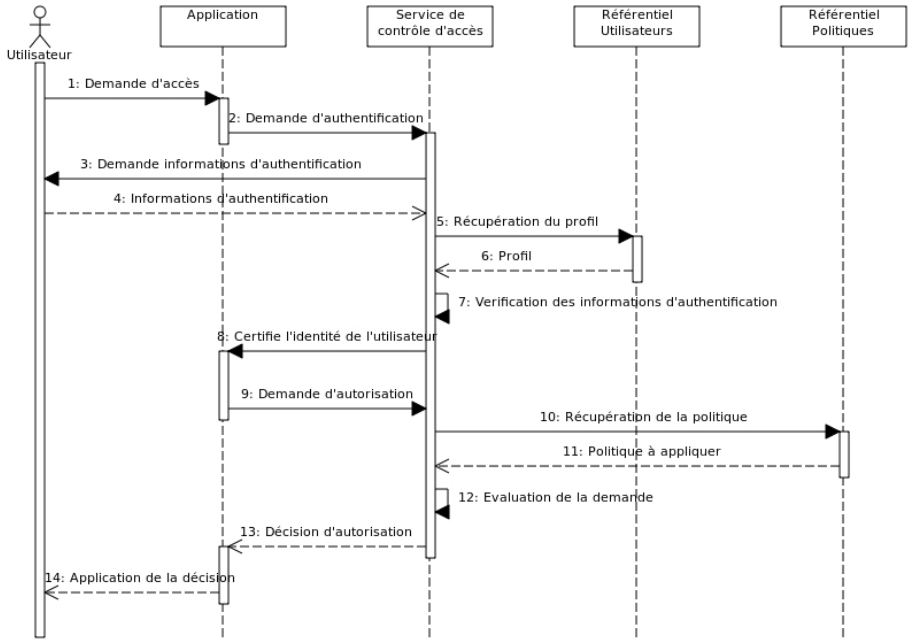

FIGURE 5.2 – Scénario général

Les échanges entre les différents composants sont représentés à la figure 5.2. Chaque étape est décrite ci-dessous :

1. L'utilisateur demande l'accès à l'application
2. L'application demande une authentification
3. Le service de contrôle d'accès demande ses informations d'authentification à l'utilisateur
4. L'utilisateur fournit ses informations d'authentification
5. Le service de contrôle d'accès récupère le profil de l'utilisateur
6. Le référentiel fournit le profil de l'utilisateur
7. Le service de contrôle d'accès vérifie les informations fournie par l'utilisateur
8. Le service de contrôle d'accès certifie de l'identité de l'utilisateur à l'application
9. L'application demande un autorisation pour cet utilisateur en précisant éventuellement certains éléments du contexte dans lequel s'effectue cette demande
10. Le service de contrôle d'accès récupère la politique de sécurité à appliquer
11. Le référentiel fournit la politique
12. Le service de contrôle d'accès prend un décision sur base de la politique, de l'utilisateur et du contexte
13. Le service d'autorisation transmet sa décision à l'application
14. L'application applique la décision reçue

Pour plus de clarté, la fonctionnalité d'audit n'a pas été représentée sur cette figure. Chaque événement (accès, tentative d'accès, requête ou réponse) sera néanmoins tracé.

Dans certains cas, l'application aura besoin d'informations supplémentaires sur l'utilisateur. Ce flux n'a pas été représenté dans le scénario ci-dessus. Le service de contrôle d'accès ayant déjà récupéré le profil de l'utilisateur, il peut être conservé afin de fournir les informations nécessaires aux applications.

5.2.2 Administration du système

L'administration du système se fait via une interface unique. Cette interface est accessible via une application d'administration. Elle comprend principalement trois fonctions, la gestion des utilisateurs, la gestion du système et la gestion des politiques de sécurité. Cette application est une application comme les autres, son accès sera donc protégé de la même manière que les autres applications.

5.3 Identification des rôles nécessaires

Afin de réaliser nos principales fonctionnalités, l'authentification, l'autorisation et l'audit, un certain nombre de rôles sont nécessaires. Nous les identifions dans cette section.

Le service d'authentification est généralement composé de trois ou quatre entités. Dans le protocole Kerberos, on retrouve un client, un serveur, un serveur d'authentification et un serveur de tickets. Pour SAML, l'Identity Provider, le Service Provider et le client (User-Agent ou ECP). Enfin, dans OpenID on retrouve également trois acteurs : l'OpenIDProvider , le Relying Party et l'User-Agent. Nous pouvons donc considérer qu'un service d'authentification est généralement composé d'un fournisseur d'identité (Identity Provider, IdP), d'un fournisseur de service (Service Provider, SP) et d'un client.

Le service d'autorisation est lui composé des quatre éléments suivants : un Policy/Authorization Information Point (PIP), un Policy/Authorization Decision Point (PAP), un Policy/Authorization Enforcement Point (PEP) et un un Policy/Authorization Administration Point (PAP). Le PIP est chargé de récupérer les informations nécessaires à une prise de décision. Ces informations concernent l'utilisateur, la ressource, l'action et l'environnement dans lequel la demande d'accès est réalisée. Ces informations doivent être connues par le PDP afin de pouvoir prendre une décision. Le PDP évalue donc la politique de sécurité en fonction des informations reçues du PIP et rend une décision d'autorisation. Le PEP protège l'accès à ses ressources en demandant une autorisation au PDP pour l'utilisateur et en appliquant cette décision. Enfin, le PAP doit écrire les politiques de sécurité à appliquer et les rendre accessibles au PDP.

Le service d'audit peut être décomposé en 3 entités : les capteurs, le gestionnaire d'événements et le journal de traces. Les capteurs doivent détecter les événements à tracer et les transmettre au gestionnaire d'événements. Le gestionnaire permet de surveiller le système et de consulter les traces de manière efficace. Enfin, le journal de traces contient toutes les traces enregistrées.

5.4 Description et localisation des composants

Dans cette section, nous examinerons les composants nécessaires afin de réaliser les différentes fonctionnalités en respectant les exigences définies et en utilisant les standards appropriés.

5.4.1 Authentification

Dans cette section, nous examinons comment offrir un service d'authentification complet répondant aux exigences que nous avons déterminées.

Authentification unique

La fonctionnalité la plus importante est d'offrir un service d'authentification unique(SSO). Il existe aujourd'hui différentes solution de SSO qu'il convient d'examiner afin de repérer laquelle pourrait convenir le mieux à notre situation.

Dans la littérature, on sépare les solutions de SSO en 2 types : les SSO basés sur le web(WebSSO) et les SSO qui ne le sont pas(Enterprise SSO). Matt Flynn distingue clairement ces deux types de SSO en deux technologies différentes.[Fly06] Pour lui le WebSSO concerne la sécurité des accès. Lorsqu'un accès est demandé, un décision est prise pour accorder ou refuser cet accès. L'ESSO en revanche permet juste d'apporter un avantage ergonomique à l'utilisateur. Un client gère l'identité de l'utilisateur auprès des différentes applications. Il n'y a donc aucun avantage au point de vue de la sécurité. Notre but est bien d'améliorer avant tout la sécurité en proposant un service de contrôle d'accès. Nous suivons donc une approche WebSSO que nous essayerons d'appliquer à un environnement contenant des applications web et non-web.

Analysons donc les différents types de WebSSO. Ils peuvent être divisés en 3 catégories : orienté Internet, Intranet ou multi-domaines. Le SSO multi-domaine peut concerner une même organisation ou des organisations partenaires, on parle alors de SSO fédéré. Les SSO non-web ne concernent en général qu'un seul domaine. Dans notre cas, le SSO sera basé sur un seul domaine. Pour le moment, seul les utilisateurs de Saint-Luc ont accès aux applications. Si un jour il est nécessaire d'ouvrir certaines applications à un partenaire, il faudra alors mettre en place une fédération. SAML apparait aujourd'hui comme le standard utilisé pour implémenter une fédération d'identité [Laa]. Shibboleth propose d'ailleurs une extension de SAML qui enrichit ses fonctionnalités de fédération d'identités [OS05]. Pour l'instant, notre solution sera basée sur un domaine unique. La problématique qui nous occupe ne nous permet pas d'utiliser directement un WebSSO puisqu'elle repose sur l'intégration d'application web et non-web. Il faudra donc combiner une approche WebSSO avec des applications web et non-web.

Les SSO peuvent être implémentés selon différentes architectures[BK11] : une architecture basée sur un courtier, sur un agent ou sur un reverse-proxy.

Dans une architecture basée sur un courtier, un serveur central authentifie les utilisateurs et leur fournit des tickets. Ces tickets pourront être utilisés pour accéder aux applications. Le protocole Kerberos décrit dans la section 4.1 en est l'exemple le plus connu.

Dans l'architecture basée sur un agent, un agent est placé en avant de chaque application ou serveur d'application. Il est chargé de traduire les informations d'authentification unique dans le protocole de l'application. L'agent peut également être placé sur le poste client. Il utilise alors une « boite à mots de passe »pour fournir à chaque application l'identifiant et le mot de passe adéquat à la place de l'utilisateur. Déporté la fonction de SSO sur le poste client est une solution qui demande peu ou pas de modifications des applications existantes mais n'apporte en revanche qu'un avantage ergonomique. La sécurité n'est en

effet aucunement améliorée.[Sal03]

Dans l'architecture basée sur un reverse-proxy, un reverse-proxy est placé au point d'entrée du système ou d'un ensemble d'application. Il filtre les demandes d'accès. Les demandes authentifiées sont transmises tandis que pour les autres, l'utilisateur est redirigé vers le serveur d'authentification afin de pouvoir authentifier ses demandes. Shibboleth peut notamment être déployé selon cette approche.

Le choix d'une architecture est difficile à faire car il dépend du type d'application et de la manière dont on veut ou peut y intégrer le système d'authentification. Les types d'applications ont été décrits dans la section 3.3. On peut distinguer les applications web et les applications non-web (C/S ou 3-tiers). Un WebSSO nécessitant obligatoirement la présence d'un navigateur tel que OpenID ou CAS n'est donc pas applicable ici. SAML en revanche propose un profil SSO basé sur un client ou un proxy. Shibboleth est basé sur SAML, il pourrait donc également être utilisé pour l'authentification. Nous nous baserons cependant sur SAML afin de rester plus générique.

Afin d'utiliser SAML, il est nécessaire que les applications puissent envoyer et recevoir des messages SAML mais également les interpréter. Restons cependant réalistes, cette intégration ne pourra pas se faire en une fois et pour certaines applications cela sera peut-être impossible. Il nous faudra donc trouver un moyen pour que ces applications puissent tout de même s'intégrer dans notre nouveau système.

Imaginons comment chacune des approches fonctionnerait avec SAML. Pour l'approche basée sur un courtier, les assertions SAML peuvent jouer le rôle de ticket. Le courtier présente alors ces assertions aux applications. Deux possibilités se présentent alors : soit l'application comprend et interprète les messages SAML, soit un agent doit traduire ces messages dans le protocole de l'application. Dans les deux cas, cela revient à proposer une approche basée sur un agent chargé de traduire ou d'interpréter les messages reçus. La combinaison des deux approches semble être un bon compromis pour intégrer progressivement l'usage de SAML. Pour intégrer SAML, il faudra donc modifier les applications afin qu'elles soient capables de demander et d'interpréter des messages SAML et ce aussi bien pour les applications web que non-web. Différentes bibliothèques, comme OpenSAML, permettent aujourd'hui d'implémenter un client SAML. Enfin, les anciennes applications qui ne pourront pas ou ne seront pas encore modifiées pourront utiliser le système actuel (Huissier) pour communiquer avec le nouveau service de sécurité. L'Huissier devra être modifié en client SAML. Il sera alors chargé de récupérer des assertions pour les applications et de les traduire dans le protocole actuellement utilisé entre ces applications et l'Huissier. L'Huissier joue donc le rôle de courtier et d'agent. L'approche reverse-proxy est quand à elle difficilement applicable aux applications non-web. De plus, elle peut difficilement être utilisée pour une décision d'autorisation plus complexe qu'une décision basée sur la seule identité de l'utilisateur au contraire des approches basées sur le courtier ou l'agent.

Nous utiliserons donc une approche basée sur un client intégré à l'applica-

tion ou une approche hybride basée sur un agent/courtier joué par l'Huissier. A présent, il est nécessaire de vérifier si les différents profils définis par SAML peuvent encore s'appliquer. Pour les applications web, le profil Web Browser SSO présenté dans la section 4.2.4 peut être appliqué tel que défini dans les spécifications. L'application web communique avec le service de sécurité via le navigateur de l'utilisateur. Pour les applications non-web, le client ou le courtier communique directement avec le service de sécurité et ne passe donc pas par un navigateur. Cette situation se rapproche du profil ECP, voyons s'il peut convenir moyennant quelques modifications si nécessaire.

Examinons les quelques différences qui existent entre le profil ECP présenté dans la section 4.2.4 et l'utilisation que nous voulons en faire. Tout d'abord, l'utilisateur n'utilise pas un client pour accéder à l'application. Il ne demande donc pas l'accès à une ressource via une requête HTTP mais directement dans l'application via différents menus, panneaux, ...On peut tout de même considérer cela comme une requête. Une fois cette requête interceptée, l'application agit comme un Service Provider en demandant une authentification via son agent (client intégré) ou son courtier (Huissier). Cette demande ne se fera pas sous forme d'un message SAML, mais sous le forme d'un appel au client ou de l'envoi d'un message à l'Huissier. La suite du scénario est ensuite respectée jusqu'à l'étape 7. En effet, le courtier ou l'agent ne transmet pas directement la réponse mais la traduit ou l'interprète avant de la transmettre. Pour cette traduction, l'Huissier joue le rôle d'agent. Le scénario ECP adapté à notre situation devient le suivant :

1. L'utilisateur fait une requête dans l'application
2. L'application demande à son agent/courtier une authentification de l'utilisateur
3. L'agent/courtier détermine l'Identity Provider
4. L'agent/courtier émet une AuthnRequest à l'Identity Provider
5. L'Identity Provider authentifie l'utilisateur si ce n'est pas encore fait
6. L'Identity Provider émet une ¡Response¿ à l'agent/courtier pour l'application
7. L'agent/courtier transmet la réponse à l'application après l'avoir traduite ou interprétée
8. L'application autorise ou refuse l'accès à l'utilisateur

Il faut remarquer que dans ces deux profils, la dernière étape consiste à refuser ou autoriser l'accès à la ressource. Dans notre cas, une demande d'autorisation devra être faite pour l'utilisateur venant d'être authentifié comme nous le verrons plus tard. Si la réponse à cette demande est positive, l'application autorisera l'accès à ses services ou ses ressources, sinon elle le refusera.

Les applications web ou non-web tiendront donc le rôle de Service Provider, le client sera joué par le navigateur, l'agent ou le courtier en fonction du profil SSO. Le rôle d'Identity Provider nécessaire aux deux profils sera joué par le service de contrôle d'accès.

Déconnexion unique(Single Logout)

SAML propose également un profil pour la déconnexion unique(Single Logout ou SLO) que nous avons présenté dans la section 4.2.4. Le profil fait intervenir deux types d'entité : les Session Participant et la Session Authority. Dans notre cas, ce sont les applications qui joueront le rôles de Session Participant et le service de contrôle d'accès qui jouera le rôle de Session Authority. Cette fonctionnalité, de même que pour l'authentification unique, nécessite la mise en place de la gestion des sessions. Nous en discuterons dans la section suivante. En effet, pour fournir une fonctionnalité de Single Logout, il est nécessaire de pouvoir contacter tous les participants de la session. Dans notre cas, ces participants sont les différentes applications dans lesquelles l'utilisateur à lancé une session. Dans le cas de l'utilisation de l'Huissier comme courtier, plusieurs applications ne seront accessibles que via celui-ci. Il conviendra donc au courtier de gérer ces applications et de pouvoir affirmer que la session de l'utilisateur est bien terminée pour chacune d'entre elles.

La gestion de sessions doit également permettre de déconnecter un utilisateur inactif. Ceci n'est nécessaire que pour les utilisateurs s'étant authentifiés avec leur mot de passe. Pour le badge, seul son retrait doit terminer une session. Pour ce faire, il est donc nécessaire de pouvoir gérer un timeout de manière centralisée pour chaque session d'un utilisateur.

L'activité peut être détectée de différentes manières. On peut utiliser les requêtes émises auprès du services de contrôle d'accès pour contrôler l'activité comme c'est le cas pour les sessions HTTP. Une autre possibilité est d'installer un client sur le poste de travail qui transmet l'activité au service de sécurité. La première possibilité ne nécessite aucune installation sur le poste de travail mais ne permet pas d'être très précis alors que la seconde permet d'être très précis mais nécessite l'installation sur un poste client. La deuxième possibilité est actuellement utilisée par l'Huissier mais pose certains problèmes lorsque plusieurs Huissier sont présents. Une dernière solution est encore possible, chaque application gère son propre timeout et avertit le service de contrôle d'accès lorsque l'utilisateur est inactif. Si l'utilisateur est inactif pour toutes les applications pour lesquelles il a ouvert une session, il est considéré comme inactif et un protocole déconnexion est lancé. Cependant, cette approche nécessite d'avoir confiance en l'application. Le service doit être certain qu'il sera avertit par chaque application si son utilisateur est inactif. Cette approche est quelque peu contradictoire avec un service de contrôle d'accès externe et centralisé. Pour le moment, la première possibilité sera conservée faute de mieux.

Gestion de sessions

Les services d'authentification unique et de déconnexion unique reposent sur la possibilité de constituer une session pour chaque utilisateur. Pour les applications web, il existe différentes manières d'implémenter cette gestion de sessions.[Oll] L'utilisation d'un cookie est l'implémentation la plus courante mais la réécriture d'URL est également possible. Pour les applications non-web, il est impossible d'utiliser les mêmes techniques. Il faut donc trouver un moyen pour

que l'Identity Provider sache que deux applications sont utilisées par un même utilisateur. Le seul point commun entre ces applications est la machine à partir de laquelle l'utilisateur les exécute. Il convient donc à l'Identity Provider d'identifier la machine derrière laquelle l'utilisateur est présent et démarre ses applications. Pour cela, il ne possède que peu d'informations. Il peut utiliser l'adresse IP et le nom de la machine qui lui est associée à un moment donné. Chaque requête reçue par l'Identity Provider pourra ainsi être associée à une machine et donc un utilisateur. Ceci nous permet d'avoir une gestion de session pour les applications web et non-web.

Récupération des informations d'authentification

L'authentification des utilisateurs doit pouvoir se faire via les moyens actuels : un couple identifiant/mot de passe ou un badge. Afin de récupérer ces informations, le service d'authentification doit pouvoir proposer un formulaire à l'utilisateur ou récupérer les informations présente sur le badge. Il est évident que la communication de ces informations devra se faire via un canal sécurisé. L'identifiant et le mot de passe peuvent être obtenus au moyen d'un formulaire HTML ou d'une boite de dialogue. Pour ce qui est du badge, l'utilisation d'un lecteur et de ses pilotes sera indispensable. Dans les deux cas, un client situé sur le poste de l'utilisateur est nécessaire pour récupérer les informations fournies par l'utilisateur et les transmettre au service de sécurité. L'introduction et le retrait du badge devront également pouvoir être détectés. L'Huissier remplit actuellement ces fonctions. Il pourrait donc être utilisé pour récupérer ces informations et ensuite les transmettre via un canal crypté au service d'authentification. L'Huissier jouera donc le rôle de client d'authentification et le service de contrôle d'accès celui du serveur.

Attributs

Dans certains cas, l'application pourrait avoir besoin d'informations supplémentaires sur l'utilisateur. Cependant, les décisions d'autorisation étant externalisées, ces demandes devraient être limitées. Le profil Assertion Query/Request présenté à la section 4.2.4 peut être utilisé à cet effet. L'application jouera le rôle de SAML Requester et le service de contrôle d'accès jouera le rôle de SAML Authority. Pour les applications utilisant un courtier(Huissier), celui-ci fournissait déjà certaines informations. Il fera donc appel au service de contrôle d'accès pour obtenir ces informations et les transmettra ensuite aux applications. Dans ce cas, c'est lui qui joue le rôle de SAML Requester.

5.4.2 Autorisation

Le service d'autorisation doit rendre des décisions d'autorisation sur base du profil de l'utilisateur, d'une politique de sécurité et du contexte dans lequel s'opère la demande d'accès. Il existe différents modèles de contrôle d'accès que nous avons présentés dans la section 2.3.4. Dans notre cas, le contrôle sera basé sur les rôles de l'utilisateur. En fonction de son rôles, l'utilisateur obtiendra une

série de droits auxquels on peut rajouter éventuellement des droits spécifiques. La majeure partie du travail a déjà été faite puisque la définition des rôles et des droits qui s'y rapportent a déjà été faite.

Nous avons présenté XACML et OAuth dans le chapitre 4. Ils proposent tous les deux un processus d'autorisation qui peut s'intégrer à SAML. OAuth est plutôt basé sur un modèle de contrôle d'accès discrétionnaire(DAC), le possesseur de la ressource autorise l'accès à une autre entité. Il ne convient donc pas dans notre contexte qui nécessite un contrôle d'accès basé sur les rôles. XACML en revanche permet de définir un politique de sécurité basée sur les rôles et également d'échanger des informations d'autorisation. De plus, ces informations d'autorisation peuvent être protégées et transportées aux travers de messages SAML comme l'explique le SAML 2.0 profile of XACML v2.0.[RL10] que nous avons déjà présenté.

Comme nous l'avons vu dans la section 5.3, un service d'autorisation nécessite différents acteurs. XACML ajoute à ceux la un gestionnaire de contexte. Voyons à présent comment ces acteurs s'articulent autour de notre architecture. Le PEP est simplement l'acteur responsable de faire appliquer la décision. Dans notre cas, ce sont simplement les applications. Le PAP doit permettre de définir les politiques de sécurité et de les rendre accessibles pour le PDP. Le PAP sera donc joué par l'application d'administration. Les PDP et PIP seront joués par le service de contrôle d'accès. Le PDP sera chargé de rendre véritablement les décisions d'autorisation et le PIP sera le composant chargé de recueillir les informations nécessaires pour rendre la décision. Le gestionnaire de contexte est chargé de traduire les requêtes et décisions d'autorisation de XACML au format de l'application mais aussi les attributs pour qu'ils soit utilisables par le PDP. La première partie correspond au rôle de l'agent ou du courtier défini plus haut. En effet, celui-ci est chargé de traduire ou d'interpréter les messages SAML. Il peut faire de même pour les messages XACML. Pour la seconde partie, le rôle de gestionnaire de contexte ne peut pas être joué par l'agent ou le courtier, celui-ci n'étant censé que servir de traducteur ou relayeur entre les applications et le service de sécurité. Cette partie sera donc intégrée au PIP afin que celui-ci transmette directement les attributs sous forme d'attributs SAML.

Dans le profil SAML de XACML présenté dans la section 4.5.3, les entités sont quelque peu différentes. L'Attribute Authority(AA) qui est chargée de lier des attributs à une identité remplace en quelque sorte le PIP. Il correspond exactement a ce que nous voulions, un PIP capable de fournir des informations sous forme d'attributs SAML. La communication entre les différents acteurs de ce profil se fait évidemment via des messages SAML.

5.4.3 Audit

Le service d'audit sera composé de différents journaux permettant d'enregistrer les événements survenus lors d'un contrôle d'accès, d'une interface pour gérer les événements et des capteurs pour les détecter comme nous l'avons présenté plus haut. Ces événements sont principalement les accès et tentatives d'accès, les requêtes et les réponses, l'authentification de l'utilisateur, l'ouverture

et la fermeture de session. Ce service sera basé sur le principe du SIEM (Security Information and Event Management) en vue de collecter ces événements et ensuite de les analyser si nécessaire. La collecte des informations peut se faire via deux modes différents [NC], un mode actif dans lequel un agent est chargé de récupérer les informations et de les envoyer au SIEM et un mode passif ou les applications envoient directement les information au SIEM sans intermédiaire.

Les applications joueront donc le rôle de capteur en mode passif. Le composant SIEM jouera le rôle du gestionnaire d'événements mais sera séparé du système de contrôle d'accès. L'application d'administration devra cependant pouvoir utiliser ses services de reporting.

5.4.4 Administration

L'administration des politiques, des utilisateurs et du système se fera par une seule et même application. Il n'y aura qu'une seule interface pour accéder aux différents types de référentiels ce qui permettra de gérer de manière centralisée le système d'information et d'ainsi éviter les incohérences. L'application d'administration devra également permettre au responsable de la sécurité de visualiser l'état du système via l'utilisation du service de reporting du SIEM. Enfin, il devra être possible de synchroniser les annuaires d'utilisateurs existants et de les utiliser pour approvisionner un référentiel unique sur lequel reposera le contrôle d'accès.

5.4.5 Gestion de la confiance

La communication d'informations de sécurité rendent les différents acteurs dépendants les uns des autres. Il est donc nécessaire qu'ils établissent une relation de confiance entre eux afin de déterminer avec quel niveau de confiance ils peuvent communiquer ou utiliser ces informations. La gestion de la confiance entre les différentes applications, le courtier et le service de contrôle d'accès représente donc un point important de la solution proposée.

Dans le cas de SAML, la confiance peut être établie par la signature des messages via XML Signature ou par un protocole réseau sécurisé tel que TLS ou IP Security Protocol[FH05]. Dans tous les cas, la confiance repose sur la cryptographie asymétrique ou cryptographie à clef publique. Cependant, l'utilisation de la cryptographie à clef publique demande une protection de l'accès à la clef privée des entités et nécessite de pouvoir lié une clef publique à une identité. Cette liaison peut être faite par un répertoire locale liant les clefs à une identité ou par l'utilisation de certificats. L'utilisation de certificats requiert une gestion du cycle de vie des certificats mais est plus évolutif qu'un répertoire local. La gestion des certificats se fait généralement par la mise en place d'une infrastructure à clef publique(PKI).

Pour les applications, elles ne doivent en principe établir une relation de confiance qu'avec une seule entité, le service de contrôle d'accès. Elles peuvent donc utiliser un répertoire local qui ne contiendra qu'une seule clef ou bien utiliser

un certificat. Le service de contrôle d'accès, peut également utiliser les deux options. Cependant, les spécifications SAML précise que si TLS est utilisé, il est obligatoire pour le serveur d'utiliser les certificats X.509 v3 pour s'authentifier auprès des clients. Nous utiliserons donc les certificats pour lier les clefs publiques à une identité.

5.4.6 Résumé des composants nécessaires

Nous pouvons donc résumer les rôles joués par les grands composants.
Pour résumer, les entités qui composent notre système sont :

- Les **applications** jouent les rôles de Service Provider, Session Participant, Policy Enforcement Point et de capteur.
- Le **navigateur** joue le rôle de User-Agent dans le Web Profil SSO.
- L'**Huissier** joue le rôle d'agent/courtier et de client d'authentification.
- Le **service de contrôle d'accès** joue les rôles d'Identity Provider, Session Authority, PDP, PIP ou AA, capteur et serveur d'authentification.
- L'**administrateur de sécurité** joue le rôle de PAP et de client du SIEM
- Le *SIEM* joue le rôle de gestionnaire d'événement

5.5 Réalisation des scénarios

Dans cette section, nous montrons comment réaliser les différents scénarios de contrôle d'accès avec les composants que nous avons identifiés.

5.5.1 Contrôle d'accès à une application

Ce scénario a déjà été défini plus haut mais nous sommes à présent capable de déterminer comment il est réalisé. Nous pouvons en réalité le séparer en deux sous-scénarios : l'authentification et l'autorisation. Nous détaillons ces sous-scénarios dans les sections suivantes.

5.5.2 Authentification de l'utilisateur

L'authentification de l'utilisateur peut être séparée en deux grandes étapes. L'authentification de l'utilisateur proprement dite et la certification de l'identité auprès des applications. L'authentification proprement dite ne doit s'effectuer qu'une seule fois puisque nous offrons un service d'authentification unique. Par contre, la certification sera faite pour chaque application démarrée. Les moyens disponibles pour authentifier l'utilisateur sont un badge et/ou un couple identifiant/mot de passe. Le service d'authentification demande donc à l'utilisateur de fournir son identité et une preuve de celle-ci. Sur base de l'identité fournie, le service d'authentification récupère le profil de l'utilisateur et vérifie que la preuve fournie par l'utilisateur est valide. Si c'est le cas, une assertion d'authentification sera produite et délivrée aux applications lorsqu'elles en font la demande.

Lorsqu'un utilisateur s'authentifie avec succès, une session sera ouverte pour lui. Comme nous l'avons expliqué plus haut, cette session sera basée sur l'adresse IP et le nom de la machine sur laquelle se trouve l'utilisateur. Cette session contiendra son profil, les assertions produite pour lui, un timeout et la liste des différentes applications auprès desquelles il aura ouvert une session et le moyen des les contacter. Tous les événements qui surviennent durant l'authentification devront être tracés en faisant appel au SIEM.

Plusieurs combinaisons sont possibles en fonction du type d'application, de la manière dont s'authentifie l'utilisateur et des éventuelles demandes précédentes d'autres applications. Dans le premier scénario (figure 5.3), l'utilisateur s'authentifie au moyen de son badge et accède à une application avec un client intégré puis à une application avec un courtier. Lorsque l'utilisateur introduit son badge, l'Huissier transmet les informations d'authentification au service de contrôle d'accès. Sur base de l'adresse ip et du nom de la machine sur laquelle se trouve l'utilisateur ainsi que des informations reçues, le service de contrôle d'accès vérifie les informations transmises et établit une session pour l'utilisateur. Lorsqu'une application émet une demande d'authentification, le service de contrôle d'accès se base à nouveau sur l'adresse ip et le nom de la machine d'ou provient la requête pour identifier l'utilisateur et ainsi lier cette demande à la session déjà établie.

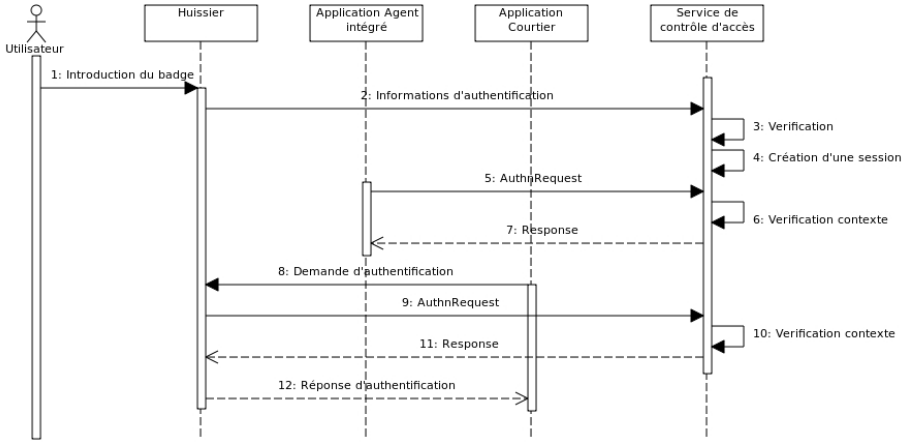

FIGURE 5.3 – Authentification par badge, application client intégré et courtier

Dans le second (figure 5.4), l'utilisateur s'authentifie avec son identifiant et son mot de passe et accède d'abord à une application web puis à une application avec un client intégré. Le scénario est assez similaire au précédent. Lorsque la première application émet une demande d'authentification, le service de contrôle d'accès vérifie si il possède déjà une session pour l'utilisateur en se basant sur l'adresse ip et le nom de la machine d'où provient la requête. Pour les application web, c'est bien l'adresse et le nom de la machine sur laquelle est exécuté le navigateur, et donc ou se trouve l'utilisateur, et pas l'adresse du serveur web qui est prise en compte. Il ne possède pas de session donc authentifie l'utilisateur à l'aide de l'Huissier. Si l'authentification réussit, il crée une session pour l'util-

isateur et émet une réponse pour l'application. Les demandes suivantes seront satisfaites en se basant sur l'ip et le nom de la machine d'où provient la requête.

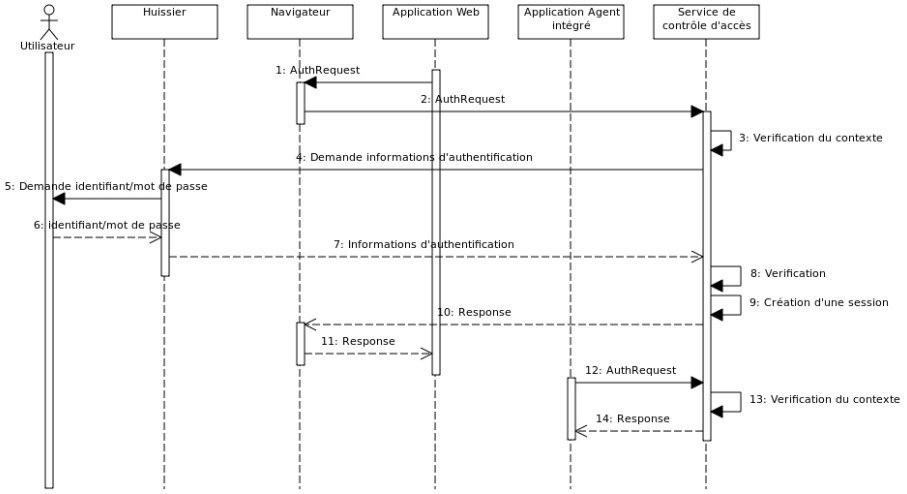

FIGURE 5.4 – Authentification par mot de passe, application web et client intégré

Pour le choix du profil, notre première idée est d'utiliser le profil Web Browser SSO. En effet, pour les applications web accessibles à l'aide d'un simple navigateur, c'est le profil adéquat. Pour les applications non-web, il n'y a pas de profil défini mais nous avons décrit une adaptation du profil ECP qui pourrait convenir.

Nous pouvons cependant remarqué que les différences entre les profils Web Browser SSO et ECP sont assez faibles. Ils utilisent tous les deux le protocole de demande d'authentification. En revanche, ils utilisent des bindings différents. De plus, dans le profil ECP, le client doit connaitre le moyen de contacter l'IdP. Le client peut donc être un navigateur s'il est capable d'utiliser le reverse-SOAP et qu'il connait le moyen de contacter l'IdP. Nous constatons également que les deux profils SSO supposent la présence d'un intermédiaire, un User-Agent ou un ECP (client ou un proxy), entre l'Identity Provider et le Service Provider. Ceci est sans doute dû au fait qu'il est nécessaire d'avoir un intermédiaire différent du Service Provider pour récupérer les informations d'authentification de l'utilisateur. Dans notre cas, c'est l'Huissier qui en a la charge. Dans sa fonction de client d'authentification, l'Huissier n'est pas un client des Service Providers mais seulement un client du service d'authentification. Il n'y a donc pas d'intermédiaire nécessaire entre le Service Provider et l'Identity Provider. On peut donc envisager une communication directe via un binding SOAP pour les applications web.

Pour les applications non-web, le profil ECP tel que nous l'avons adapté ne nécessite plus vraiment l'utilisation du reverse-SOAP (PAOS). En effet, dans notre scénario adapté, la première requête HTTP n'est pas présente. Un binding SOAP peut donc être utilisé.

En résumé, les applications utilisant un client intégré peuvent communiquer directement avec le service de contrôle d'accès en utilisant un binding SOAP. Les applications utilisant l'Huissier comme agent et courtier ne communiquent pas directement avec le service de contrôle d'accès mais n'émettent et n'interprètent pas de messages SAML. L'Huissier en donc une communication directe pour l'échange des messages SAML avec le service de contrôle d'accès. Il peut lui aussi utiliser un binding SOAP. Les applications web peuvent utiliser un binding HTTP ou, si le navigateur le supporte et que l'application connait le moyen de contacter le service de contrôle d'accès, un binding reverse-SOAP.

Nous utiliserons le profil Web Browser SSO avec un binding HTTP POST pour les applications web. Celui-ci étant très largement utilisé, il est important que l'Identity Provider puisse le supporter. Pour les applications non-web, nous utiliserons un binding SOAP pour la communication entre le client intégré ou l'Huissier jouant le rôle d'agent et de courtier et le service de contrôle d'accès.

Voyons à présent les messages échangés. Les messages échangés entre l'Huissier et les applications sont propres au protocole défini entre eux. Pour les échanges entre l'Huissier et le service de contrôle d'accès, il faut prévoir deux scénarios. Dans le premier scénario, l'utilisateur insère son badge ou introduit son mot de passe avant d'utiliser une application. C'est donc à l'Huissier de contacter le service de contrôle d'accès en lui transmettant les informations d'authentification de l'utilisateur. Dans le second scénario, c'est le service de contrôle d'accès qui demande à l'Huissier de récupérer les informations d'authentification auprès de l'utilisateur. Dans les deux cas, la communication devra se faire de manière sécurisée.

Voyons à présent les messages SAML utilisés. Il est conseillé d'utiliser SSL ou TLS pour échanger les `AuthnRequest` et les `Response`. Pour le profil Web Browser SSO, l'Identity Provider doit pouvoir établir que c'est bien le Service Provider qui reçoit la `Response` à l'autre extrémité. Pour cela, il doit vérifier que l'élément `AssertionConsumerServiceURL` ou `AssertionConsumerServiceIndex` est bien le Service Provider auquel doit être envoyée la `Response`. Vu que nous utilisons le binding HTTP POST, les assertions contenues dans la réponse devront être signées. L'émetteur de l'`AuthnRequest` doit être précisé dans l'élément `Issuer` avec l'identifiant unique du Service Provider. Si une erreur se produit, une `Response` ne contenant aucune assertion doit être émise. Elle contiendra le code d'erreur correspondant à celle qui est survenue. Si il n'y a pas d'erreur, elle doit contenir au moins une assertion contenant un `AuthnStatement`. Il faudra préciser l'émetteur pour chaque assertion. Un `AuthnStatement` doit préciser l'utilisateur auquel elle se rapporte et préciser la manière dont il a été authentifié. Il faut également préciser à quelle requête se rapporte la réponse. Un élément `AudienceRestriction` doit préciser l'identifiant du Service Provider.

Pour le profil ECP, certaines précisions sont faites pour l'échange des messages SOAP. Les remarques concernant les messages SAML, proprement dit, sont les mêmes que pour le profil Web Browser SSO. Nous garderons donc ces recommandations pour nos échanges avec le profil ECP adapté.

5.5.3 Décision d'autorisation

Les applications doivent obtenir des décisions d'autorisation pour leur utilisateur. Cette demande a toujours lieu après que l'utilisateur ai été authentifié, le service de contrôle d'accès doit donc posséder une session pour cet utilisateur. Pour rendre une décision d'autorisation, en plus du profil de l'utilisateur, il est nécessaire de récupérer des informations sur le contexte et la politique de sécurité à appliquer. Les informations sur le contexte doivent être fournies par l'application dans la demande d'autorisation. La politique de sécurité doit être récupérée dans le référentiel des politiques de sécurité. La décision sera rendue sous la forme d'une assertion conservée dans la session de l'utilisateur. Cette assertion sera ensuite transmise à l'application. Tous les événements survenus seront également tracés.

Nous pouvons distinguer 2 scenarii. Dans le premier scénario (figure 5.5) une application utilisant un courtier émet une demande d'autorisation et en reçoit la réponse. Dans le second scénario (figure 5.6), une application utilisant un client intégré (web ou non-web) demande une décision d'autorisation et reçoit la décision en réponse. Les applications n'ont pas besoin de passer par l'intermédiaire de l'Huissier pour communiquer avec le service de contrôle d'accès. Excepté cela, le scénario est le même.

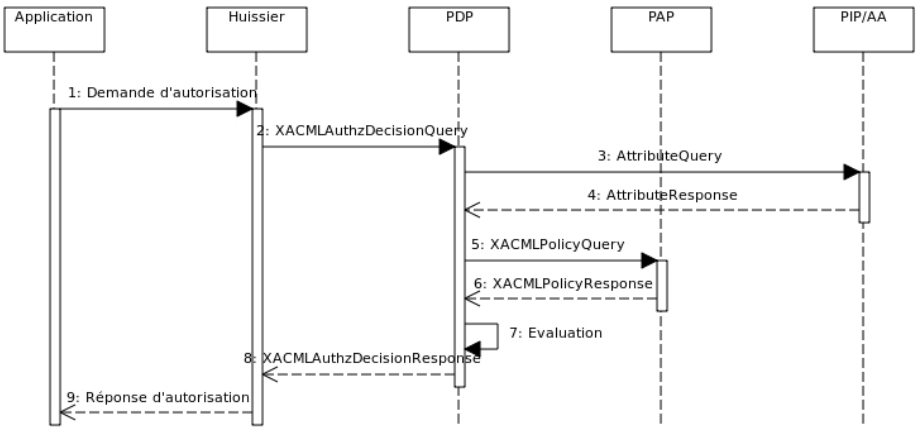

FIGURE 5.5 – Décision d'autorisation : application avec courtier

Nous pouvons utiliser le profil Assertion Query/Request pour les trois types de requêtes et de réponses. Ce profil implémente le protocole de requête et de demande d'assertions en utilisant le binding SOAP. Il n'y a pas de surprise particulière. Il est conseillé que les deux entités s'authentifient et que l'intégrité du message soit assurée par une signature ou par un autre mécanisme supporté par le binding. La confidentialité doit également être garantie pour les demandes attributs et si possible pour les demandes de politiques et de décisions. On peut donc utiliser TLS ou XML Signature et XML Encryption. On préférera TLS qui permet l'authentification mutuelle, l'intégrité et la confidentialité en un seul mécanisme. On peut également ajouter que la communication entre PDP et PIP ou PAP peut se faire directement si ils sont déployés dans une seule application.

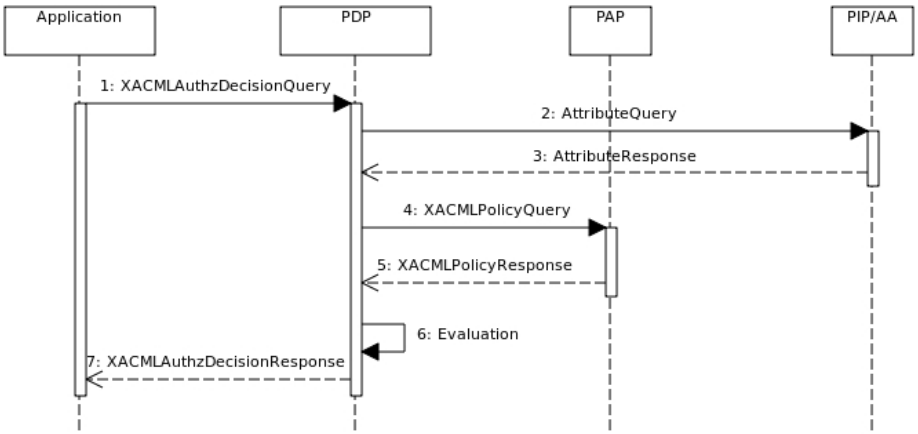

FIGURE 5.6 – Décision d'autorisation : application web ou avec client intégré

Le protocole de requête et de demande d'assertions exige que les messages contiennent l'élément Issuer. Passons maintenant aux messages proprement dit.

La demande `XACMLAuthzDecisionQuery` est de type `XACMLAuthzDecisionQueryType`, lui même une extension du type SAML `RequestAbstractType`. La requête sera donc composée d'un élément `Issuer` et des attributs `ID`, `Version` et `IssueInstant`. A cela on ajoute une requête XACML. Les autres attributs ou éléments de la demande permettent de spécifier la politique et les attributs sur lesquels doivent être pris la décision. Dans notre cas, ce n'est pas à l'application de décider de cela, ils sont donc ignorés. La requête XACML contient différents attributs et précise si la réponse doit contenir les politiques appliquées et si une décision combinée est demandée. Dans les deux cas, ce n'est pas nécessaire. Les attributs d'une requête XACML sont regroupés par catégorie. Chaque attribut est composé d'un identifiant et d'une valeur.

La réponse est une instance d'une `Response` SAML. Elle contient le statut de la réponse et au moins une assertion s'il n'y a pas eu de problème. L'élément `Issuer` doit être utilisé pour respecter le protocole de demande/requête d'assertions. Une assertion `XACMLAuthzDecision` doit être inclue dans la réponse et le statut SAML de la réponse doit correspondre au statut XACML de la réponse XACML. L'assertion doit contenir un `Issuer`, la version et l'`IssueInstant` mais pas de `Subject`. Elle contient également un `Statement` de type `XACMLAuthzDecisionStatementType`. Ce type de `Statement` doit contenir une réponse XACML. Une réponse XACML est simplement constituée d'un élément résultat pour chaque ressource. Cet élément résultat contient une décision (Permit, Deny, Indeterminate ou NotApplicable).

5.5.4 Déconnexion unique

Un service de déconnexion unique est proposé à l'utilisateur. De cette manière, il peut fermer toutes ses sessions applicatives en une seule fois. Pour proposer

ce service à l'utilisateur, le service de contrôle d'accès doit ouvrir une session pour chaque utilisateur s'étant authentifié de manière correcte. Il associe un index à chaque session. Il est conseillé d'utilisé un petit entier dont la valeur est choisie au hasard parmi un intervalle. Cet index est transmis aux applications dans l'assertion d'authentification. L'application utilisera cet index lorsqu'elle émettra une requête de déconnexion.

La déconnexion unique peut être demandée par l'utilisateur auprès d'une application ou directement auprès du service de contrôle d'accès. Le service de contrôle d'accès peut également initier le scénario lorsque l'utilisateur est inactif par exemple. On peut considérer que le retrait du badge ou la demande de déconnexion à l'Huissier est une demande initiée par l'utilisateur.

Dans le premier scénario (figure 5.7), l'utilisateur retire son badge. Il est ensuite déconnecté d'une application web et d'une application avec un client intégré.

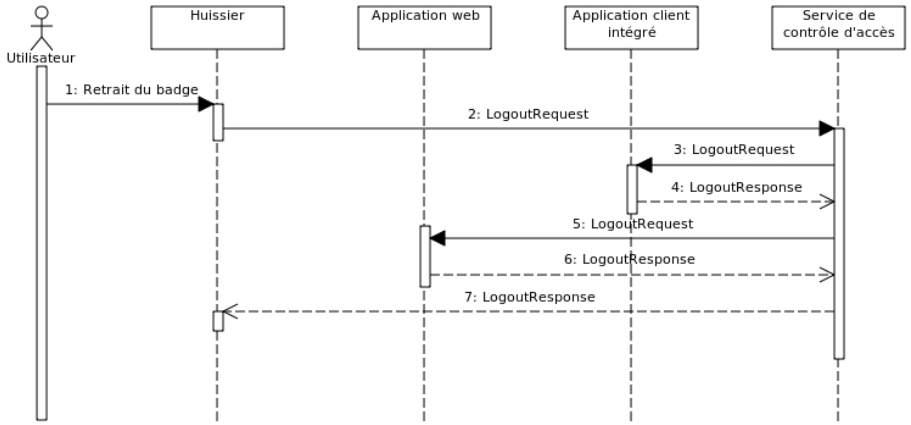

FIGURE 5.7 – Single Logout par retrait du badge

Dans le second scénario (figure 5.8), l'utilisateur demande à l'Huissier de fermer sa session. L'Huissier émet donc une `LogoutRequest` en tant que client d'authentification comme le ferait n'importe quelle application. Le service de contrôle d'accès identifie la session et envoie une demande de déconnexion à toutes les applications participantes à la session. Ce sont des applications utilisant l'Huissier comme courtier. L'Huissier va donc recevoir une demande de déconnexion pour chaque application participante à la session dont il est le courtier. Lorsque chaque application confirme que la session locale de l'utilisateur a été fermée par l'envoi d'un `LogoutResponse`, le service de contrôle d'accès peut confirmer à l'Huissier que l'utilisateur à bien été déconnecté de toutes les sessions. L'Huissier confirme alors à l'utilisateur qu'il a bien été déconnecté.

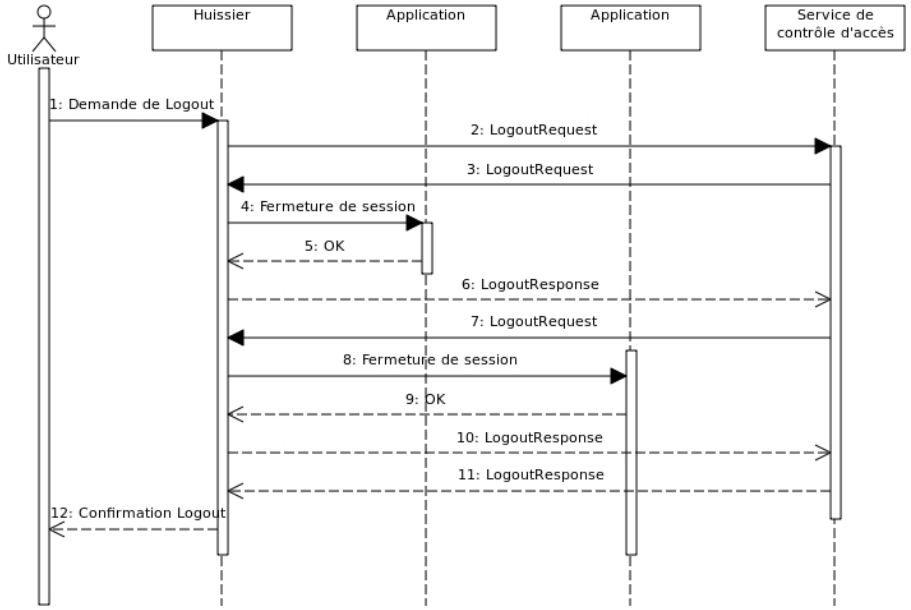

FIGURE 5.8 – Single Logout pour les applications utilisant un courtier

5.5.5 Demande d'informations supplémentaires sur l'utilisateur

Une demande d'informations supplémentaires sur l'utilisateur peut être nécessaire pour certaines applications. Cependant, il faudra définir les informations auxquelles les applications peuvent accéder. Le scénario n'est pas détaillé, il consiste juste à une demande et une réponse.

Le profil demande l'utilisation de SOAP et il est préférable que le demandeur et le répondeur s'authentifie mutuellement. L'émetteur de chaque message, demande ou réponse, doit être précisé dans l'élément `Issuer`. Il est préférable que l'intégrité des messages soit assurée. Nous utiliserons une connexion SSL ou TLS afin de protéger l'intégrité et la confidentialité des messages et d'authentifier les deux entités.

5.6 Déploiement de la solution

Maintenant que nous avons déterminé l'architecture et montré que différents scenarii pouvaient être réalisés, nous pouvons examiner de manière générale les étapes nécessaires pour déployer la solution. Pour cela nous nous inspirons des différentes étapes proposées dans [Sem05].

Tout d'abord, la solution repose sur deux sources de données pour proposer un service de contrôle d'accès, une base de donnée centrale des utilisateurs et une politique de sécurité des droits d'accès logiques unifiée et centralisée.

La première étape sera d'identifier les différentes bases de données utilisateurs, les procédure de gestion utilisateurs, la structure de l'organisation et les éventuelles réglementation à respecter. Pour Saint-Luc, nous savons qu'il existe une base de donnée Huissier qui constitue déjà une unification des principales ressources de données utilisateur.

La seconde étape consiste à déployer un système de provisionnement. Le but est de pouvoir utiliser efficacement les procédures de gestion d'utilisateurs afin de ne pas créer d'incohérence entre les différents comptes utilisateurs présents dans le système. Il est également nécessaire de mettre en place le système d'audit et les moyens de gérer les utilisateurs et leurs rôles. Concrètement, cela revient à développer l'application d'administration des utilisateurs et à leur attribuer leurs rôles. L'identification des rôles et leur attribution a déjà été faite à Saint-Luc. Le système actuel se base déjà sur des rôles. La mise en place du service d'audit revient à mettre en place la gestion des événements par le déploiement d'un Security Information and Event Management (SIEM) s'il n'en existe pas encore. Nous pouvons ajouter à cela la mise en place d'une infrastructure à clef publique.

La troisième étape est la dernière étape avant le déploiement proprement dit du service de sécurité. Il faut donc constituer les sources d'informations nécessaires. La base de données centrale des utilisateurs sera donc créée. Elle doit être l'unique autorité pour le système de contrôle d'accès. Il faut également traduire la politique de sécurité des droits d'accès logiques existante en un ensemble de politique XACML afin de constituer un politique unifiée et centralisée.

La quatrième étape est le déploiement du service de contrôle d'accès. Maintenant que nous avons une base centralisée des utilisateurs et une base centralisée des politiques, nous pouvons nous baser dessus comme source d'information pour l'authentification, l'autorisation et les attributs. Le service devra offrir les interfaces nécessaires pour offrir ses services au applications. De plus, il devra pouvoir utiliser les bases de données des utilisateurs et des politiques et les services du SIEM afin de tracer les événements. Il faudra également lui générer un certificat.

La cinquième étape consiste à adapter l'Huissier et les applications afin qu'ils puissent utiliser les services de contrôle d'accès qui leurs sont proposés. L'Huissier devra être adapté pour de devenir un client d'authentification afin de fournir les informations d'authentification transmises par l'utilisateur et de détecter l'introduction ou le retrait du badge. Il devra également être adapté afin de devenir un courtier pour certaines applications. Les applications devront être adaptée progressivement pour utiliser le service de contrôle d'accès en utilisant SAML. Il faudra également générer un certificat pour chaque application et pour l'Huissier.

5.6.1 « SAMLiser »l'Huissier

Afin que l'Huissier puisse émettre et recevoir des messages SAML, il doit être transformé en client SAML et XACML. Différentes bibliothèques clientes existent dont la plus connue est OpenSAML [Shi], développée par Shibboleth afin

d'implémenter son WebSSO basé sur SAML. Elle est disponible pour les langages Java et C++. Les fonctionnalités proposées actuellement pourront facilement être traduites en requêtes d'authentification, d'autorisation ou d'attributs.

5.6.2 « SAMLiser »les applications

Le système est composé de différents types d'applications. Il ne sera pas possible de les modifiées toutes en même temps. Il y aura donc une cohabitation pendant un certain temps entre le système actuel et la solution proposée. C'est d'ailleurs essentiellement à cet effet que l'Huissier est adapté afin de devenir un agent et un courtier. Pour modifier les applications, il faudra également utiliser la bibliothèque cliente OpenSAML. La définition d'une interface commune qui serait implémentée par les applications pourrait être un moyen de réduire le travail de développement. Pour les applications impossible à modifier, il ne sera pas possible qu'elles utilisent le service de contrôle d'accès. Au mieux, elle pourront utiliser le service d'authentification et obtenir des attributs supplémentaires via un agent.

Chapitre 6

Considérations de sécurité

Différentes analyses ont été faites sur les protocoles SAML [SH04], [AA08], [Gro03]. Comme expliqué dans [AA10], la définition d'un standard permettant une grande interopérabilité à pour conséquence d'offrir un nombre important de combinaisons possibles provenant des champs optionnels dans les messages, de l'utilisation de canaux sécurisés ou de l'utilisation du chiffrement ou de la signature sur les messages. Les recommandations qui accompagnent le standard permettent de se prémunir des principaux pièges mais ne permettent pas de s'assurer qu'aucun piège n'est présent. La manière dont est implémenté SAML peut donc amener à certaines failles comme celle survenue en 2008 sur SAML-based SSO for Google Apps. A notre niveau, l'identification des principales menaces et principaux contre-scénarios permet de limiter les risques pour peu que notre solution puisse y répondre.

Les menaces qui portent sur la solution concernent les informations échangées (sécurité des communications) et les entités qui les échangent (sécurité du système) [Ra03]. La sécurité des communications peut être divisée en trois catégories : l'intégrité, la confidentialité et l'authentification mutuelle. La sécurité du système consiste à protéger les machines et leurs données. Les machines ne doivent être utilisées que par les personnes autorisées et pour un usage approprié sans gêner les utilisateurs légitimes. Notre solution de contrôle d'accès a pour but de se prémunir contre les accès non-autorisés ou non-appropriés aux applications et donc aux données par la mise en œuvre d'une politique de sécurité des droits d'accès logiques. L'utilisation de notre solution est donc la solution à la problématique de sécurité du système.

Afin d'examiner la sécurité des communications, il faut supposer la présence d'un attaquant, comme expliqué dans [SH04], capable d'intercepter les messages, de crypter ou décrypter des messages s'il possède la clef de chiffrement, de construire un nouveau message et d'envoyer des messages créés ou interceptés.

Dans l'ITSEC (Information Technology Security Evaluation Criteria) on parle d'attaques passives ou actives.[CCC] L'attaque passive consiste seulement à lire les informations échangées tandis que l'attaque active suppose que l'attaquant émet des informations sur le réseau. Sur base de ceci et des considérations de sécurité de SAML[FH05] et XACML[Ris10a], nous pouvons iden-

tifier les contre-scenarii suivants :

- l'écoute
- le déni de service
- l'attaque par rejeu
- l'insertion de messages
- la suppression de messages
- la modification de messages
- l'attaque man-in-the-middle

Nous pouvons à présent évaluer notre solution pour chacun des scénarios identifiés.

6.1 L'écoute

Dans ce scénario, l'attaquant se contente d'un rôle passif. Il écoute les différents messages à travers le réseau. Seul la confidentialité des données est en jeu. Les données peuvent être des informations sur l'utilisateur, sur les demandes et réponses d'autorisation et sur les politiques de sécurité. La divulgation des informations sur l'utilisateur ou sur les attributs contenus dans des requêtes et réponses peut être une atteinte à la vie privée tandis que la divulgation d'informations sur la politique de sécurité peut faciliter la recherche d'une brèche dans cette politique par l'attaquant.

L'utilisation de SSL et TLS comme nous le faisons permet de se prémunir de ce genre d'attaque en assurant la confidentialité des messages échangés.

6.2 Le déni de service

Le déni de service consiste à rendre inaccessible le système à un utilisateur légitime. Pour cela, l'attaquant peut surcharger le service de sécurité, si bien qu'il ne lui est plus possible de répondre aux demandes légitimes. Ce type d'attaque est difficilement évitable. Le risque peut cependant être minimisé en authentifiant le client grâce à l'utilisation de TLS ou SSL ou en demandant la signature des requêtes. Dans notre solution, nous utilisons TLS. Seuls les messages pour lesquels le client sera authentifié seront acceptés. De plus, le service de contrôle d'accès n'est ouvert qu'au réseau intranet de Saint-Luc, le risque d'une attaque d'envergure est donc plutôt limitée. Un mécanisme de gestion de charge peut également être mis en place afin de limiter les effets d'une telle attaque.

6.3 L'attaque par rejeu

L'attaque par rejeu consiste à répéter un message valide de manière frauduleuse. L'utilisation de SSL ou TLS limite les possibilités de capture du message par un attaquant. Il est donc difficile à l'attaquant de pouvoir rejouer un message. De plus, chaque message possède un identifiant unique. La réception de

deux messages ayant le même identifiant sera donc repérée. Le moment où a été émise l'assertion est également précisé. Le récepteur sera donc en mesure de connaitre le moment où l'attaque par rejeu a eu lieu.

6.4 L'insertion de messages

Un message peut être inséré dans le flux de données échangées entre deux entités. Une fausse réponse d'autorisation peut conduire, par exemple, à un accès non-autorisé. L'utilisation de l'attribut InResponseTo rend plus difficile l'insertion d'une réponse car l'attaquant doit intercepter la requête et obtenir son identifiant. De plus, l'authentification de l'émetteur via l'utilisation de SSL ou TLS permet de se prémunir d'une insertion d'une entité inconnue.

6.5 La suppression de messages

La suppression de message consiste à supprimer un message afin qu'il n'atteigne pas son destinataire. Pour chaque requête, un réponse doit être envoyée même dans le cas ou une erreur est survenue. L'absence de réponse à une requête signifie donc qu'une suppression a eu lieu.

6.6 La modification de messages

La modification des messages peut amener de nombreuses erreurs ou incohérences. L'utilisation de TLS ou SSL permet de protéger l'intégrité des messages et donc de s'assurer que le message reçu est bien celui qui a été envoyé.

6.7 L'attaque man-in-the-middle

Cette attaque consiste à se placer entre deux entités afin d'écouter ou de modifier des messages. Ces deux cas de figure ont déjà été traités. Cette attaque est évitée par l'authentification mutuelles des entités communicantes. Cette authentification est réalisé grâce à l'utilisation de certificats et de SSL ou TLS. La validité des certificats utilisés sera vérifiée par chacune des parties.

Chapitre 7

Conclusion

La protection du système d'information est devenue aujourd'hui une préoccupation importante. Le milieu hospitalier a besoin de pouvoir protéger son système d'information qui est pour lui un outil essentiel. Le contrôle de l'accès aux applications est donc devenu une exigence cruciale qu'il convient de pouvoir faire de manière efficace et cohérente. C'est dans cette optique que nous avons cherché à proposer un service de contrôle d'accès externe et centralisé.

Les exigences et fonctionnalités d'une organisation sont nombreuses. L'authentification, la gestion des autorisations et l'audit sont les fonctionnalités centrales et nécessaires au contrôle d'accès. De plus, le service doit pouvoir s'intégrer dans le système existant afin de pouvoir être utilisé par les différentes applications présentes.

A partir de ces fonctionnalités principales, nous avons effectué un état de l'art des principaux standards et protocoles existant en matière d'authentification et d'autorisation. Nous avons ainsi étudié des protocoles d'authentification comme OpenID, SAML ou Kerberos. Nous avons ensuite examiné OAuth et XACML qui permettent de gérer les autorisations. Enfin, nous avons observé l'architecture logique des solutions utilisant ces différents protocoles.

A partir des exigences et des standards existants, nous avons pu proposer un service de contrôle d'accès basé sur SAML et XACML pour l'échange d'informations de sécurité. Nous avons ainsi décrit les différents composants et les rôles qu'ils jouaient dans notre solution. Nous avons ensuite expliqué comment ces différents composants interagissent entre eux afin de répondre aux principaux scénarios tels que l'authentification, la demande d'une autorisation ou la déconnexion. Enfin, nous avons décrit les grandes étapes nécessaires pour mettre en place notre solution.

Pour finir, nous avons identifié les principales menaces qui s'appliquent à notre service de sécurité. Nous avons décrit précisément de quelle manière notre solution répondait à ces attaques.

Bibliographie

[AA08] L. Compagna J. Cuellar L. Tobarra A. Armando, R. Carbone. Formal analysis of saml 2.0 web browser single sign-on : Breaking the saml-based single sign-on for google apps, 2008.

[AA10] L. Compagna G Pellegrino A. Armando, R. Carbone. Automatic security analysis of saml-based single sign-on protocols, 2010.

[AJ] Colin Boyd Audun Jøsang, Roslan Ismail. A survey of trust and reputation systems for online service provision.

[BK11] E. Bertino and T. Kenji. *Identity Management : Concepts, Technologies, and Systems*. 2011.

[CCC] CECA-CEE-CEEA.

[Col] Jean-Noël Colin. Sécurité et fiabilité des systèmes informatiques.

[ea05a] S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. `http://docs.oasis-open.org/security/saml/v2.0/`, 2005.

[ea05b] S. Cantor et al. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. `http://docs.oasis-open.org/security/saml/v2.0/`, 2005.

[ea05c] S. Cantor et al. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. `http://docs.oasis-open.org/security/saml/v2.0/`, 2005.

[FH05] Eve Maler Frederick Hirsch, Rob Philpott. Security and privacy considerations for the oasis security assertion markup language (saml) v2.0, 2005. `http://docs.oasis-open.org/security/saml/v2.0/`.

[Fly06] Matt Flynn. Single sign-on ; multiple confusion points, 2006. `http://360tek.blogspot.com/2006/11/single-sign-on-multiple-confusion.html`.

[Gro03] Thomas Groß. Security analysis of the saml single sign-on browser/artifact profile, 2003.

[HL10] E. Hammer-Lahav. The oauth 1.0 protocol, 2010.

[Inta] L'interopérabilité : définition et enjeux. `http://www.gralon.net/articles/internet-et-webmaster/logiciel/article-l-interoperabilite---definition-et-enjeux-3750.htm`.

[Intb] Internet2. About shibboleth. `http://shibboleth.internet2.edu/about.html`.

[ISO09] ISO/IEC. ISO/IEC 27000 : Information technology, security techniques, information security management systems, overview and vocabulary, 2009.

[Laa] Miska Laakkonen. Identity federation and identity providers.

[Lema] LemonLDAP : :NG. Lemonldap : :ng portal. http://lemonldap-ng.org/documentation/1.0/portal.

[Lemb] LemonLDAP : :NG. Lemonldap : :ng presentation. http://lemonldap-ng.org/documentation/presentation.

[Maz] Drew Mazurek. Cas protocol. http://www.jasig.org/cas/protocol.

[Mic] Microsoft. Automation. http://msdn.microsoft.com/en-us/library/dt80be78.aspx.

[Mic03] Sun Microsystems. A brief introduction to xacml, 2003.

[MM05] Paul Madsen and Eve Maler. Saml v2.0 executive overview, 2005.

[Na05] C. Neuman and al. The kerberos network authentication service (v5), 2005.

[NC] R. Picard et E. Thuiller N. Cherriere, G. Montassier. Les siem (security information and event management) : Gestion de la sécurité centralisée.

[OAu] http://old.sitepen.com/labs/code/ttrenka/oauth/oauth-sequence.png.

[Oll] Gunter Ollmann. Web based session management best practices in managing http-based client sessions. http://www.technicalinfo.net/papers/WebBasedSessionManagement.html.

[Ope07] OpenID. Openid authentication 2.0, 2007.

[OS05] Pascal Aubry Olivier Salaün, Florent Guilleux. Fédération d'identités et propagation d'attributs avec shibboleth, 2005.

[Poi] Laurent Poinsot. Politiques et modèles de sécurité. Cours Sécrypt.

[Ra03] E. Rescorla and al. Guidelines for writing rfc text on security considerations. RFC, 2003. http://www.ietf.org/rfc/rfc3552.txt.

[Ris10a] Erik Rissanen. extensible access control markup language (xacml) version 3.0, 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf.

[Ris10b] Erik Rissanen. Xacml v3.0 core and hierarchical role based access control (rbac) profile version 1.0, 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cs-01-en.pdf.

[RL10] Erik Rissanen and Hal Lockhart. Saml 2.0 profile of xacml, version 2.0, 2010.

[Sal03] Olivier Salaün. Introduction aux architectures web de single sign-on. 2003.

[Sem05] Radovan Semančík. Enterprise digital identity architecture roadmap. Technical report, nLight, 2005.

[SH04] Jakob Skrivera and Steffen M. Hansen. Using static analysis to validate saml protocols. Master's thesis, Informatics and Mathematical Modelling, Technical Uni- versity of Denmark, 2004.

[Shi] Shibboleth.

[Shi00] R. Shirey. RFC 2828 - internet security glossary, 2000.

[VM] Julien Marchal Vincent Mathieu, Pascal Aubry. Single sign-on open-source avec cas (central authentication service). http://www.esup-portail.org/consortium/espace/SSO_1B/cas/jres/cas-jres2003-article.pdf.

[Wik] Wikipedia. Système d'information. http://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27information.